P. Cabanis

Electronique digitale

DUNOD

Electronique digitale

par
Pierre CABANIS

Avec la collaboration de
Emmanuel BERNIER

FORMATION CONTINUE

Dunod

Avant-propos

Si le vingtième siècle a connu la révolution industrielle, le vingt-et-unième s'annonce comme celui où l'électronique digitale va transformer tous les secteurs de l'activité humaine, qu'il s'agisse des télécommunications, des automatismes, de l'audio-visuel et de la « hifi », de l'électro-ménager ou de l'automobile par exemple (mais cette liste pourrait être prolongée à l'infini).

Nous avons conçu cet ouvrage en pensant à tous ceux qui vont participer à cet essor de l'électronique digitale, qu'ils soient étudiants ou déjà engagés dans la vie professionnelle, et notamment

— aux élèves des stages de formation permanente ou des cours d'enseignement technique,

— aux adultes qui pressentent l'arrivée de ces techniques dans leur métier et à tous ceux qui participent à la conception de produits, d'outillages ou de montages, en bureau d'étude ou en service de méthodes par exemple.

Pour que cet ouvrage puisse leur servir de guide personnel, nous avons recherché la simplicité, la progressivité et l'aspect pratique :

Simplicité : Aucune connaissance préliminaire n'est requise sinon le bagage scolaire du premier cycle. S'il vaut mieux savoir comment fonctionne un transistor en commutation on pourra, à défaut, imaginer qu'il s'agit d'un interrupteur commandé par une source de courant.

Progressivité : Chaque notion, chaque outil de raisonnement, chaque fonction de l'électronique digitale est présenté dans un ordre tel qu'il suffit de se référer aux paragraphes précédents pour comprendre cette connaissance nouvelle.

Aspect pratique : Tout apport de connaissance est accompagné de son application. Pour cela nous faisons très souvent appel aux nombreux circuits intégrés que chacun peut se procurer auprès d'un détaillant. A la suite de la plupart des chapitres des exercices et des applications permettent de mettre immédiatement en pratique les notions acquises et de s'entraîner ainsi à la conception d'ensembles de plus en plus complexes.

*
* *

Pour tirer le meilleur parti de ce livre, nous conseillons à nos lecteurs de suivre patiemment chacun de ses chapitres, dans l'ordre dans lequel ils sont présentés et de s'appliquer à résoudre les exercices et à réaliser les montages proposés.

Ils verront alors que ces techniques sont finalement d'une simplicité désarmante et débouchent pourtant sur un monde infini d'applications où presque tout reste encore à inventer.

© BORDAS, Paris, 1985
ISBN : 2-04-015941-X

"Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de l'auteur, ou de ses ayants-droit, ou ayants-cause, est illicite (loi du 11 mars 1957, alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. La loi du 11 mars 1957 n'autorise, aux termes des alinéas 2 et 3 de l'article 41, que les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective d'une part, et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration."

Table des matières

CHAPITRE 1 — **Les portes électroniques** 1

Introduction 1
1. L'élément d'information 1
2. Traitement des informations binaires 3
3. Un outil pratique de raisonnement : les tableaux de Karnaugh 6
4. Autres relations entre deux données binaires 8

CHAPITRE 2 — **Les circuits logiques complexes** 13

Introduction 13
1. Les tableaux de Karnaugh .. 13
2. Algèbre de Boole 14
3. Réalisation des circuits logiques 16
4. Les portes à «collecteur ouvert» 19
5. Aiguillages de données 20
6. Système binaire pur 22
7. Simplification du réseau d'aiguillages 23
Exercices 24

CHAPITRE 3 — **Les bascules** 25

Introduction 25
1. La bascule bistable R.S ... 25
2. Monostables 32
3. Les multivibrateurs astables . 35
Exercices 37

CHAPITRE 4 — **Les compteurs et la numération binaire** 40

1. Compter avec des bascules... mais jusqu'où 40
2. Le code *BCD* 41
3. Le code interne des machines électroniques 46
4. Le comptage 47
5. Application : réalisation d'un fréquencemètre 56
Exercices 59

CHAPITRE 5 — **Le multiplexage et les registres à décalage** 60

1. Principe général du multiplexage 60
2. Registres à décalage 65
3. Décalage à gauche 71
Exercices 72

CHAPITRE 6 — **Les mémoires à accès aléatoire** 74

1. Mémoires et rangement 74
2. Structure interne des circuits matriciels de mémoires 85
3. Chronogrammes de lecture et d'écriture 86
Exercices 87

CHAPITRE 7 — **Les opérations de calcul arithmétique** 90

Introduction 90
1. L'addition 90
2. La soustraction 93
3. La comparaison 94
4. Unité de calcul arithmétique et logique 94
5. La multiplication de nombres binaires 95
6. Opération en *BCD* 96
Exercices 100

CHAPITRE 8 — **La programmation des opérations de traitement** 101

Introduction 101
1. Analyse des opérations élémentaires requises pour une addition 101
2. Réalisation pratique d'un additionneur de deux nombres de huit chiffres 102

CHAPITRE 9 — **Entrée des données : le clavier** 113

Introduction 113
1. Codage direct des touches .. 113

2. Codage des touches par multiplexeur 114
3. Codage des caractères alphanumériques 116
4. Gestion d'un clavier alphanumérique 118
Exercice commenté 119

CHAPITRE 10 — **Au cœur du microprocesseur** 123

Introduction 123
1. Architecture d'un microprocesseur 123
2. L'unité centrale de traitement (UCT) 126
3. Vocabulaire 131
4. Nombre de cycles requis pour une instruction 131
5. Rôle particulier de la pile lifo 131

CHAPITRE 11 — **Le langage de programmation des microprocesseurs** 134

Introduction 134
1. Instructions de chargement dans l'accumulateur 134
2. Opérations arithmétiques et logiques portant sur deux valeurs 136
3. Opérations d'incrémentation et de décrémentation 140
4. Opérations de décalage 141
5. Les opérations de rangement ou de transfert des données . 141
6. Récapitulatif sur les instructions de transfert et de traitement des données. Introduction des mnémoniques 142
7. Les instructions de prise en compte des conditions 143
8. Sauts inconditionnels 144
9. Fonctionnement de la pile du 6502 145
Exercices 145
Annexe 149

CHAPITRE 12 — **Les interfaces d'un micro-ordinateur** 150

Introduction 150
1. Ports d'entrée/sorties en parallèle 150
2. Interface parallèle/série 152
3. Les interruptions 155
Annexe 1 : Rôle des registres de contrôle d'un PIA 158
Annexe 2 : Liaison d'un ACIA avec un modem 160

CHAPITRE 13 — **Les automates programmables** 162

Introduction 162
1. Architecture d'un automate programmable 162
2. Les capteurs 163
3. Les actionneurs 167
4. Le Grafcet 168
5. Notions de séquence 170
6. Aléas 172
7. Les temporisations 174
8. Modes de fonctionnement .. 174
9. Synchronisation des séquences 176
10. La programmation d'un automate 177
Exercices 180

Réponses aux exercices 181

CHAPITRE 1

Les portes électroniques

INTRODUCTION

L'électronique digitale, appelée encore électronique numérique, s'est développée autour de la notion la plus simple qui soit : exprimer toute information avec des 0 et des 1.

Parallèlement, des techniques mécaniques, hydrauliques, pneumatiques se sont développées autour de cette même idée. Elles ont leur application dans des domaines particuliers. Mais l'électronique présente de tels avantages de rapidité, de fiabilité, de faible encombrement et de faible coût, que c'est dans son domaine qu'ont vu le jour les systèmes les plus évolués et les plus sophistiqués de traitement de l'information.

Comment? C'est ce que nous allons voir ensemble, en commençant par manier les deux unités de l'électronique digitale : le « zéro » et le « un ».

1. L'ÉLÉMENT D'INFORMATION

Le *« digit »*, terme qui vient de l'anglais, désigne tout élément d'information que l'on peut compter sur les doigts. Ainsi le système numérique habituel (décimal) que nous utilisons contient-il dix digits, de 0 jusqu'à 9, et il n'est pas étonnant que ce nombre corresponde aussi au nombre de doigts de nos deux mains.

En fait l'électronique digitale réduit à deux le nombre de digits : il ne s'agira plus que du 0 et du 1. C'est donc une numération binaire (deux éléments) et il faudrait parler de digits binaires.

Les anglo-saxons ont trouvé une abréviation de *« binary digit »* (digit binaire) en créant le mot *« bit »* que nous emploierons souvent dans cet ouvrage car il est devenu d'usage international.

Ainsi toute l'électronique digitale utilise deux éléments d'information, le bit « 0 » et le bit « 1 ».

1.1 Types d'informations traitées

Les digits binaires 0 et 1 ne vont pas servir qu'à créer un système numérique nouveau (que nous étudierons plus loin), c'est pourquoi nous avons parlé d'informations et pas uniquement de nombres ou de chiffres. En utilisant des codes adaptés nous verrons aussi qu'il est possible de traiter les caractères alphabétiques et les différents signes ou ponctuations de nos langages humains.

Bien plus, ces deux bits peuvent servir à coder des phénomènes physiques, telles que les fluctuations d'une onde sonore, si bien que l'électronique digitale est en passe de supplanter l'électronique analogique dans de nombreux domaines (la « Hi-Fi par exemple).

Mais avant d'aborder des exemples de ce dernier type, on peut facilement imaginer comment coder deux états d'un système physique simple. Par exemple :

- un réservoir est vide (0) ou plein (1),
- un commutateur est ouvert (0) ou fermé (1),
- un moteur est à l'arrêt (0) ou en marche (1),
- une porte est fermée (0) ou ouverte (1).

Dans ce dernier exemple on voit bien que le codage revêt une part d'arbitraire. On pourrait aussi bien imaginer que la porte fermée soit « codée » 1 et la porte ouverte « codée » 0.

Dans tous ces exemples, et bien d'autres que vous pourrez imaginer, le *codage* en binaire de l'information à traiter est une affaire de définition. L'essentiel, une fois que cette définition est établie, est de s'y tenir.

1.2 Traduction électronique de l'information

La solution la plus généralement adoptée en électronique, pour signifier le 0 ou le 1 est de faire apparaître sur un fil conducteur une tension.

La plupart du temps cette tension sera de 0 volt (ou voisine de 0 volt à quelques dixièmes de volt près) pour le « zéro » et de 5 volts pour le « un ». Mais il ne s'agit pas là d'une solution générale.

Certains circuits admettront des tensions négatives pour le 0 et des tensions positives (pouvant dépasser 10 volts) pour le 1. De plus en plus, cependant, la tendance est à la normalisation : 0-5 volts; c'est pourquoi nous traiterons essentiellement de ce type de solution.

Lorsqu'on utilise une information traduite par une tension sur un conducteur, on court le risque de voir cette tension chuter, ce qui va rendre l'information douteuse; aussi ne s'étonnera-t-on point de voir qu'un des composants les plus utilisés en électronique digitale est le transistor. Alimenté sous 5 volts et monté en émetteur commun, on le fait fonctionner en régime saturé ou bloqué. *Saturé :* la tension de collecteur indiquera le « zéro »; *bloqué :* cette tension indiquera le « un ».

1.3 Exemple de la traduction électronique d'une information

Si l'on veut saisir concrètement une information et la traduire dans un système électronique, il faut inventer un *capteur.* L'interrupteur manuel est le plus simple et on verra plus loin de nombreux cas d'utilisation (par exemple pour recueillir l'information provenant d'un clavier de machine à écrire). Mais on s'efforce souvent de capter de façon automatique l'information.

Un exemple classique est constitué par le couplage d'une source lumineuse et d'une cellule photo-électrique (*cf.* figure 1. 1).

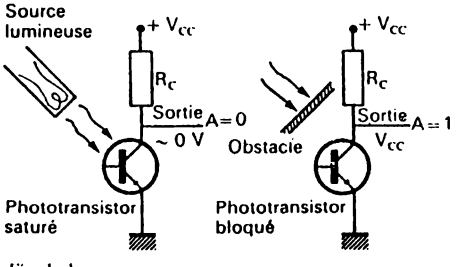


Fig. 1. 1

En temps normal le faisceau lumineux orienté vers le phototransistor entraîne la saturation de ce dernier. La tension prélevée sur le collecteur de ce transistor est donc pratiquement nulle (à la tension de saturation près entre collecteur et émetteur).

Si le faisceau lumineux est coupé par le passage d'une personne par exemple, le transistor passe à l'état bloqué et la tension de sortie s'élève brutalement à 5 volts.

L'information A, en sortie, passe ainsi de 0 à 1. Cet état 1 ne dure que le temps de passage de la personne devant le faisceau lumineux. Tout de suite après A revient à 0.

1.4 Exploitation de l'information

La présence ou l'absence d'une tension sur un conducteur n'est pas un phénomène évident. Un voltmètre pourrait le mettre en évidence, mais c'est un instrument relativement coûteux et trop subtil lorsqu'il s'agit de rendre manifeste deux valeurs seulement.

On pensera alors aux systèmes lumineux ou auditifs.

Si l'on reprend l'exemple précédent (dispositif de surveillance d'un portillon) on peut imaginer une alarme lumineuse s'allumant lorsque le faisceau lumineux de surveillance a été « coupé ».

Il faut tout d'abord transformer l'information passagère qui en résulte en une information persistante. C'est ce que l'on appelle mettre en « mémoire » cette information. Nous en étudierons en détail le principe au chapitre 3. Toutefois, en attendant, nous utiliserons l'un des composants classiques étudiés dans ce chapitre : *la bascule* D. Le circuit intégré 7474 en contient deux, mais une seule suffira.

Le câblage de ce circuit doit se faire suivant le schéma de la figure 1. 2. L'alimentation (+ 5 V) est branchée sur la borne 14 et la masse sur la borne 7.

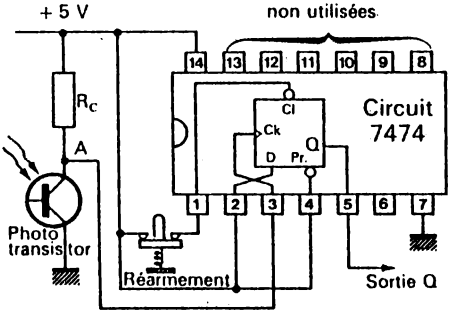


Fig. 1. 2

Les bornes 2 et 4 sont également reliées à la borne 5 V. Un bouton-poussoir de coupure est raccordé à la borne 5 V d'une part et à la borne 1 d'autre part.

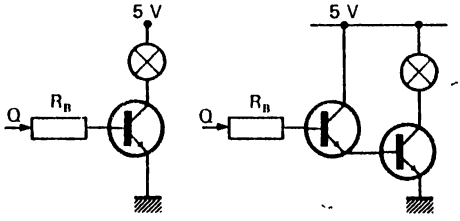
L'information A est recueillie sur la borne 3. Enfin la sortie Q est représentée par la borne 5.

Ayant au préalable appuyé sur le bouton poussoir de réarmement, tant que l'information A est à 0, Q reste à 0.

Dès qu'une impulsion de tension se produit sur A, le potentiel de Q passe à une valeur voisine de 5 V et y demeure après le retour de A à 0.

Pour remettre Q à 0 il faudra appuyer à nouveau sur le bouton-poussoir. Le circuit sera ainsi réarmé pour une nouvelle alarme.

L'information A étant ainsi mémorisée sur la borne Q, il est possible de créer une alarme lumineuse. Suivant la puissance adoptée pour l'ampoule de l'alarme le montage nécessitera un ou deux transistors (*fig.* 1. 3).



Montage pour voyant lumineux de faible puissance.

Montage Darlington pour ampoule type «lampe de poche».

Fig. 1. 3

2. TRAITEMENT DES INFORMATIONS BINAIRES

Toute information, une fois saisie, peut être stockée, reprise, transformée, comparée et associée à d'autres informations. Ces opérations sont relativement faciles à réaliser grâce aux circuits intégrés proposés par de nombreux constructeurs.

2.1 Traitements effectués sur une information unique

Une information unique peut-être :
- mise en mémoire
- amplifiée (ou régénérée)
- inversée.

La mise en mémoire sera étudiée au chapitre 3.

2.1.1 L'amplification

L'amplification restitue la même information (A → A). Elle est souvent utilisée en électronique digitale pour régénérer des signaux dont la tension aurait tendance à s'affaiblir.

Cette amplification peut être obtenue par deux transistors montés en série, ou, plus souvent par un amplificateur opérationnel intégré dont on utilise la sortie non inverseuse.

Les symboles usuels, de cette amplification sont présentés en figure 1. 4.

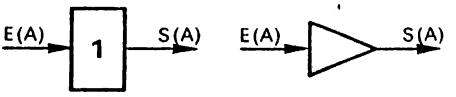


Fig. 1. 4 *a* *b*

Le symbole *a* correspond aux normes européennes, le symbole *b* aux normes américaines.

2.1.2 Inversion

En utilisant la sortie inverseuse d'un même amplificateur opérationnel, ou en montant un transistor en émetteur commun sur la sortie A de l'information, on obtient ce que l'on peut appeler l'inversion de l'information :

$$0 \rightarrow 1$$
$$1 \rightarrow 0$$

On dit aussi que l'on transforme l'information en son *complément*. Il s'agit du complément à 1 de l'information.

Le complément de A est symbolisé par $\overline{A}$ (qui se lit «A barre»). Si A = 1, $\overline{A}$ = 0.

Quant aux symboles des circuits électroniques d'inversion, ou inverseurs, ils sont présentés sur la figure 1. 5 :

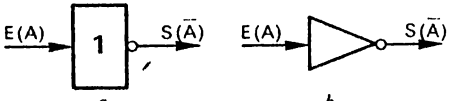


Fig. 1. 5 *a* *b*

le symbole *a* correspond aux normes européennes,

le symbole *b* correspond aux normes américaines.

Notez que dans les deux cas on a repris le symbole de l'amplificateur suivi d'un petit rond qui est le symbole de l'inversion.

2.2 Traitements associant deux informations

De nombreux circuits électroniques digitaux fournissent une information de sortie à partir de plusieurs informations d'entrée. Nous commencerons par étudier les circuits comportant deux entrées et une sortie.

Les deux entrées A et B pouvant être à 0 ou à 1, toutes les configurations possibles à l'entrée sont :

A = 0 B = 0
A = 0 B = 1
A = 1 B = 0
A = 1 B = 1

Si, en regard de ces différentes valeurs de A et B on inscrit la valeur prise par la sortie S, on obtient un tableau appelé « table de vérité » du circuit.

Chaque loi d'association différente de A et B conduit à la réalisation d'un circuit différent. Tous ces circuits sont appelés des *portes logiques*. La porte logique est caractérisée par la loi d'association des données d'entrée pour créer la valeur de sortie.

2.2.1 La porte « OU » ou « OR »

Si A *ou* B (ou les deux à la fois) prennent la valeur 1, alors S vaut 1. Telle est la loi d'association de la porte « OU » (« OR » en anglais).

La table de vérité correspondante est

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Cette table de vérité fait ressortir que, pour les trois premières lignes, S résulte de l'addition de A et de B :

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$

Mais les lois de l'addition ne sont plus respectées dans le cas où A et B valent 1.

C'est pourquoi cette loi d'association est appelée « addition logique », à ne pas confondre avec l'addition mathématique. On utilisera cependant le signe « + » pour caractériser cette relation, réservant alors le mot « plus » à l'addition mathématique de deux valeurs binaires :

A + B signifie A OU B

Pour illustrer une telle opération logique, prenons, par exemple, le cas d'un citoyen français qui veut aller en Angleterre.

Pour franchir la frontière, il doit présenter soit son passeport soit sa carte d'identité. Nous définirons les valeurs binaires A, B et S de la façon suivante :

A = 1 : il dispose d'un passeport (sinon A = 0)
B = 1 : il dispose d'une carte d'identité (sinon B = 0)
S = 1 : il est autorisé à franchir la frontière.

Le résultat S découle bien de l'addition logique A + B.

Dans le cas où A = 1 et B = 1, on ne refoulera pas notre citoyen français pour autant!

La réalisation électronique d'une telle porte ne présente pas de difficulté particulière. La figure 1, 6 en donne un exemple. La présence des diodes est indispensable pour éviter les court-circuits éventuels entre A et B.

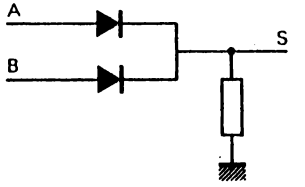


Fig. 1. 6

Les fabricants de circuits intégrés proposent des portes prêtes à l'emploi : le circuit 7432 contient quatre portes OU à deux entrées.

Le symbole européen de ces portes est présenté en figure 1. 7 *a*).

Les États-Unis utilisent le symbole de la figure 1. 7 *b*).

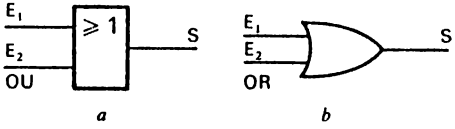


Fig. 1. 7

Par suite de la plus grande diffusion de la documentation américaine, et peut-être de la meilleure lisibilité de ce symbole, c'est ce deuxième symbole qui est de loin le plus utilisé, et c'est lui que nous adopterons dans la suite de cet ouvrage, désignant la fonction qu'il réalise par les termes de « OU » ou de « OR ».

2.2.2 La porte « ET » ou « AND »

La loi d'association de A *ET* B peut être exprimée sous la forme suivante :

le résultat S de l'association A ET B ne peut prendre la valeur 1 que si A et B sont tous deux égaux à 1.

La table de vérité qui en découle est la suivante : .

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Cette table de vérité fait ressortir une parfaite identité entre le ET logique et les lois de la multiplication

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

C'est pourquoi cette relation logique est aussi appelée produit logique de A et de B. Cette fois le produit logique est bien identique au produit arithmétique mais il ne faudra pas faire de confusion entre le ET logique (qui suggère une notion d'addition) et l'addition logique (qui correspond au OU).

A × B (ou A . B) signifie A ET B

Reprenant un exemple « administratif » on peut appliquer cette loi d'association aux conditions que doit remplir un citoyen français pour être autorisé à participer à un vote.

Cette autorisation (S) lui sera donnée s'il présente sa carte d'électeur (A) et une pièce d'identité (B) :

$$S = A \cdot B$$

S'il ne dispose pas d'un des deux documents, il ne peut voter. *A fortiori* s'il n'a ni l'un, ni l'autre.

La réalisation électronique d'une porte ET peut utiliser les mêmes composants que ceux de la porte OU, dans une autre disposition (*fig.* 1. 8).

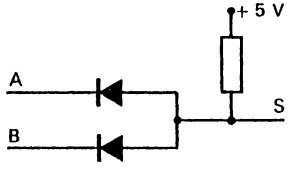


Fig. 1. 8

Il sera souvent plus pratique de faire appel à un composant tel que le circuit intégré 7408 qui contient quatre portes « ET » à deux entrées dans un seul boîtier à quatorze broches.

Le symbole européen de la porte ET est présenté sur la figure 1. 9 *a*.

Les États-Unis utilisent le symbole de la figure 1. 9 *b*. Ici encore, c'est ce deuxième symbole que nous utiliserons dorénavant (porte AND).

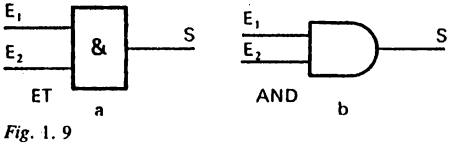


Fig. 1. 9

2.2.3 La porte « OU-NON » ou « NOR »

Le résultat d'un OU entre A et B peut fort bien être inversé, ce qui donnera la table de vérité suivante :

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

S n'est « vraie » que si A et B sont faux.

Comme cette relation prend le contre-pied de la relation OU on l'appelle OU-NON. En anglais elle devient la relation « NOR » par contraction de non-or.

Il n'y a pas de signe logique spécifique à cette relation. On se contentera d'écrire que le résultat du OU-NON entre A et B est le complément du résultat du OU, en mettant une barre unique sur l'ensemble de la relation A + B

$S = \overline{A + B}$ signifie S = A OU-NON B

Si un train circule (S) tous les jours sauf les jours fériés (A) et les dimanches (B), on peut exprimer ceci sous forme de relation OU-NON :

$$S = \overline{A + B}$$

La réalisation d'une porte électronique OU-NON peut se déduire de celle d'une porte OU : il suffit d'y ajouter un transistor monté en émetteur commun. Il est toutefois prudent de remplacer les diodes par des résistances pour ne pas imposer une tension base-émetteur trop forte (*fig.* 1. 10).

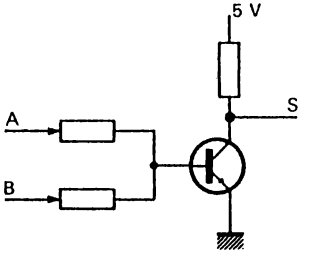
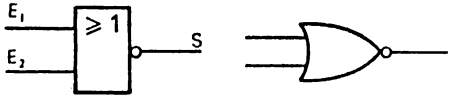


Fig. 1. 10

Le circuit intégré 7402 propose quatre portes NOR (OU-NON) à deux entrées dans un seul boîtier à quatorze broches.

Les symboles utilisés pour représenter ces portes conservent le symbole de la porte OU suivi du petit cercle, symbole de l'inversion.



a : OU-NON *b* : NOR

Fig. 1. 11

Ils sont présentés sur la figure 1. 11.

Nous utiliserons dorénavant le symbole américain.

2.2.4 Porte «ET-NON» ou «NAND»

En inversant le résultat de l'opération « ET » on obtient la table de vérité suivante :

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

S n'est « faux » que si A et B sont « vrais ».

Pour écrire cette relation, on écrira, grâce au symbole « barre », que S est le complément (l'inverse) de A . B :

$\overline{S = A \cdot B}$ signifie S = A ET-NON B

Si un conducteur roule à plus de 60 km/h (A) sur une route alors qu'un panneau indicateur limite la vitesse autorisée à 60 km/h (B), ce conducteur n'est pas dans son droit (S = 0). L'absence de panneau indicateur (ou de législation) ou le fait qu'il roule au-dessous de 60 km/h, justifie sa conduite (S = 1).

La réalisation, en électronique, d'une porte ET-NON s'inspirera du schéma de la porte ET (voir *fig.* 1. 12).

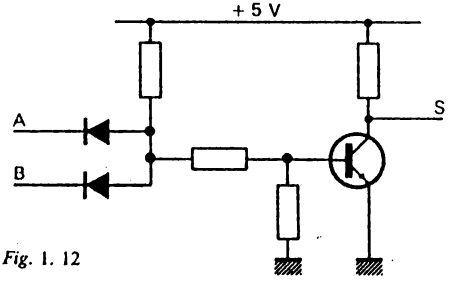
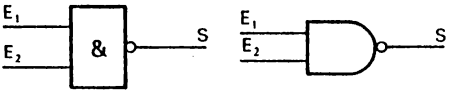


Fig. 1. 12

Le circuit intégré 7400 propose quatre portes NAND (ET-NON) à deux entrées en un seul boîtier à 14 broches.

Quant aux symboles représentant ces portes, ils reprennent les symboles de la porte ET ou AND, suivis du petit cercle indiquant l'inversion de la donnée de sortie (*fig.* 1. 13).



a : ET-NON *b* : NAND

Fig. 1. 13

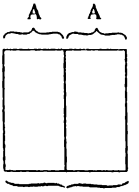
3. UN OUTIL PRATIQUE DE RAISONNEMENT : LES TABLEAUX DE KARNAUGH

Pour aller plus loin dans l'analyse des différentes relations qui peuvent exister entre des variables binaires, il est indispensable de se doter d'un outil à la fois simple et très pratique; il s'agit du tableau de Karnaugh, du nom de son inventeur.

3.1 Principe de représentation de variables binaires sur un tableau de Karnaugh

Une variable binaire ne prend que deux valeurs : 0 ou 1. Si on appelle A sa valeur à un moment donné, A n'étant pas connu, on peut dire que cette variable ne pourra avoir, comme valeur, que A ou $\overline{A}$.

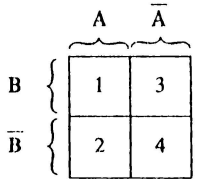
Si l'on représente arbitrairement le domaine de variation de A par la surface interne à un carré, on peut partager ce carré en deux zones : la zone où la variable prend la valeur A et la zone où cette variable prend la valeur $\overline{A}$:



domaine des valeurs
d'une variable binaire

Considérons maintenant une nouvelle variable binaire pouvant prendre les valeurs B et $\overline{B}$. Son domaine de variations peut être représenté par le même carré que le domaine de variation de A. En choisissant une autre ligne de démarcation entre

le domaine de B et celui de $\overline{B}$ on peut déterminer, dans ce carré, tous les cas possibles :



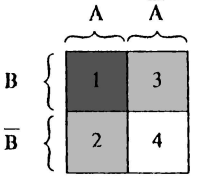
la zone 1 correspond à A et B
la zone 2 correspond à A et $\overline{B}$
la zone 3 correspond à $\overline{A}$ et B
la zone 4 correspond à $\overline{A}$ et $\overline{B}$

3.1.1 Représentation de la fonction OU

Ce tableau très simple permet de visualiser les différentes fonctions que nous venons d'étudier.

Ainsi, pour représenter S = A + B (fonction OU), il suffira de mettre en grisé tout le domaine où la première variable prend la valeur A et tout le domaine où la deuxième prend la valeur B, pour obtenir les trois zones 1, 2 et 3.

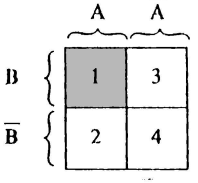
On voit encore pourquoi le OU logique correspond à ce qu'on appelle l'*union* de l'ensemble A et de l'ensemble B



3.1.2 Représentation de la fonction «ET»

Pour représenter le produit logique A . B correspondant à la fonction A ET B, il faut mettre en grisé sur le tableau la zone correspondant à A et B, excluant toutes celles où figureraient les valeurs de $\overline{A}$ ou de $\overline{B}$.

Il n'y a donc que la zone 1 qui corresponde à cette fonction :



On voit aussi pourquoi on dit que le ET logique correspond à *l'intersection* de l'ensemble A et de l'ensemble B.

Dans ce même ordre d'idée, la zone 2 correspond à A . $\overline{B}$, la zone 3 à $\overline{A}$. B et la zone 4 à $\overline{A}$. $\overline{B}$.

3.1.3 Utilisation des tableaux de Karnaugh en algèbre binaire (ou algèbre de Boole)

Si nous reprenons le tableau de Karnaugh de la fonction A + B, en partant du principe qu'en binaire tout ce qui n'est pas noir est blanc (tout ce qui n'est pas S est $\overline{S}$), nous pouvons dire que tout ce qui ne fait pas partie de A + B correspond à $\overline{A + B}$.

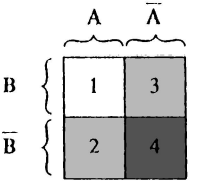
C'est donc la zone 4 qui correspond à $\overline{A + B}$.

Mais nous venons de voir que cette même zone 4 correspond aussi à $\overline{A}$. $\overline{B}$.

On peut en déduire une relation importante de l'algèbre binaire :

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

On peut alors étudier la fonction $\overline{A} + \overline{B}$, en la figurant en grisé sur le tableau de Karnaugh :



Pour constater aussitôt qu'elle correspond au complément de A . B (case 1).

Donc $\overline{A} + \overline{B} = \overline{A \cdot B}$.

Ces deux relations fondamentales constituent ce que l'on appelle le théorème de Morgan, du nom de son inventeur, Augustus de Morgan).

3.1.4 Relations entre algèbre de Boole et électronique

Il n'y a aucune raison que ce qui est vrai en algèbre ne le soit pas lorsqu'il s'agit de traiter des valeurs binaires à travers des circuits électroniques.

Nous avons vu que la porte OU permettait d'obtenir la relation S = A + B.

Supposons (*fig.* 1. 14) que nous ne disposions que de portes ET et d'inverseurs. Nous pouvons

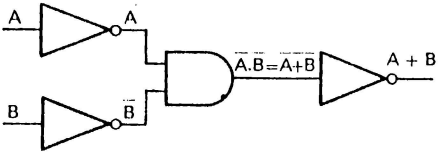


Fig. 1. 14

cependant obtenir la valeur S, en inversant tout d'abord A et B, en réalisant ensuite la fonction $\overline{A} \cdot \overline{B}$ (équivalente à $\overline{A + B}$) et en inversant enfin cette dernière valeur.

Noter à ce propos que l'inverse de $\overline{A + B}$ pourrait s'écrire $\overline{\overline{A} + \overline{B}}$, mais qu'il est plus simple de l'écrire $A + B$: en algèbre binaire une double inversion restitue la donnée d'origine.

4. AUTRES RELATIONS ENTRE DEUX DONNÉES BINAIRES

Pour identifier toutes les lois d'association possibles de deux valeurs binaires, il est commode de se référer au tableau de Karnaugh.

Nous avons déjà vu le cas où une seule des quatre cases 1, 2, 3 ou 4 vérifie la relation. Ceci correspond à la fonction ET.

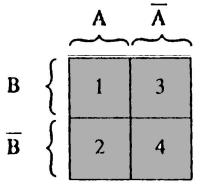
Nous avons vu aussi que lorsque trois cases vérifient la relation nous avons affaire à une fonction OU.

Évoquons maintenant les autres cas.

4.1 Pseudo-fonctions

Dans le tableau suivant c'est l'ensemble des quatre cases 1, 2, 3, 4 qui est en grisé. S = 1 quelles que soient les valeurs de A ou de B.

On pressent qu'il n'est vraiment pas nécessaire de disposer d'une porte pour créer cette fonction : il suffit de relier S aux 5 volts de l'alimentation.

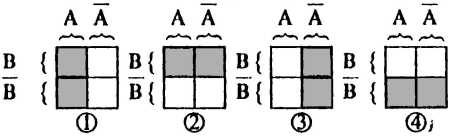


Pourtant il nous faut remarquer que ce qui paraît évident sur le tableau de Karnaugh l'est moins lorsque les relations se présentent sous la forme :

$$S = A + \overline{A}$$
$$S = B + \overline{B}$$
$$S = A + B + \overline{A}\,\overline{B}$$
$$S = A + \overline{B} + \overline{A}B$$
$$S = \overline{A} + B + A\overline{B}$$
$$S = \overline{A} + \overline{B} + AB$$

Mais si, pour chacune de ces relations, vous établissez le tableau de Karnaugh vous constaterez que, chaque fois, elles sont équivalentes à S = 1.

— Un deuxième cas de « pseudo-relations » correspond aux figures ci-dessous :



le cas 1 correspond à A, le cas 2 à B, le cas 3 à $\overline{A}$ et le cas 4 à $\overline{B}$.

Si l'analyse, par le tableau de Karnaugh, d'une fonction plus ou moins complexe de deux variables, ramène à un de ces quatre cas, on pourra en déduire qu'il n'est nul besoin de disposer de portes à deux entrées pour obtenir S puisqu'une seule variable est en jeu. S sera identique à cette variable ou à son inverse (complément)

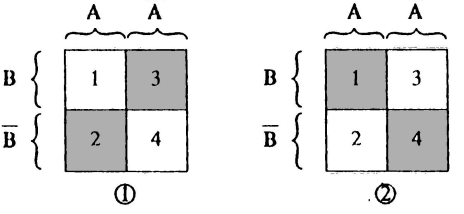
Vous pouvez vérifier que la relation

$$S = (\overline{A} + B) \cdot (\overline{A} + \overline{B})$$

fait partie de l'un de ces cas.

4.2 Le « OU EXCLUSIF »

Il reste alors deux cas que nous n'avons pas encore étudiés. Ils correspondent aux tableaux de Karnaugh suivants :



La relation définie par les zones en grisé du deuxième tableau est complémentaire de celle définie par le premier. On ne va donc étudier que ce premier tableau, une inversion du résultat permettant d'obtenir la deuxième relation.

Notons tout de suite que l'on peut traduire la relation du tableau ① en algèbre binaire, sous la forme

$$S = A \cdot \overline{B} + \overline{A} \cdot B \text{ (ou } S = A\overline{B} + \overline{A}B)$$

et nous pouvons en dresser la table de vérité, en nous aidant pour cela des deux valeurs intermédiaires $A\overline{B}$ et $\overline{A}B$.

A	B	$\overline{A}$	$\overline{B}$	$A\overline{B}$	$\overline{A}B$	S
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

d'où le tableau simplifié

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

S est égal à 1 lorsque A et B ont des valeurs complémentaires. Chaque fois que A et B sont identiques (0 ou 1) S prend la valeur 0.

On appelle cette fonction le *OU-EXCLUSIF* (ou EXOR en anglais) car, semblable au OU elle exclut toutefois la possibilité que A et B soient ensemble égaux à 1.

Elle caractérise ce qu'on appelle un dilemme : choix entre deux solutions qui exclut la possibilité de les adopter ou de les rejeter toutes deux ensemble.

Plus prosaïquement, les électriciens ont depuis longtemps réussi à câbler un circuit répondant à cette fonction : c'est le « va-et-vient » comportant deux interrupteurs séparés qui permettent, chacun de leur côté, d'allumer la même lampe (*fig.* 1. 15).

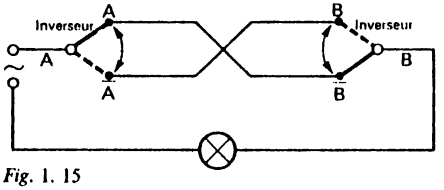


Fig. 1. 15

La lampe ne s'allume que pour les positions $A\overline{B}$ ou $\overline{A}B$ des interrupteurs-inverseurs. Si nous appelons S = 1 la lampe allumée, nous avons bien :

$$S = A\overline{B} + \overline{A}B.$$

En électronique il est amusant d'essayer de réaliser la fonction OU-EXCLUSIF à l'aide de composants discrets, en s'inspirant des réalisations précédentes et en utilisant des transistors pour inverser les valeurs de A et de B.

On obtient alors le schéma de la figure 1. 16, mais on peut aussi utiliser des circuits tout prêts, tel que le circuit 7486 proposant quatre portes OU-EXCLUSIF indépendantes dans un seul boîtier à 14 bornes.

Remarque : Si l'on ne dispose pas du circuit 7486, on peut aussi réaliser cette porte avec deux portes ET, une porte OU et deux inverseurs, en calquant le schéma sur la relation $S = A\overline{B} + \overline{A}B$ (voir *fig.* 1. 17).

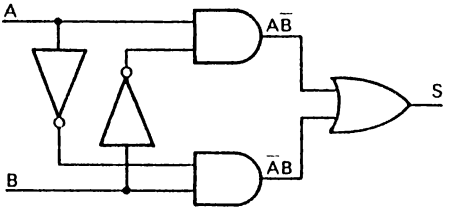


Fig. 1. 17

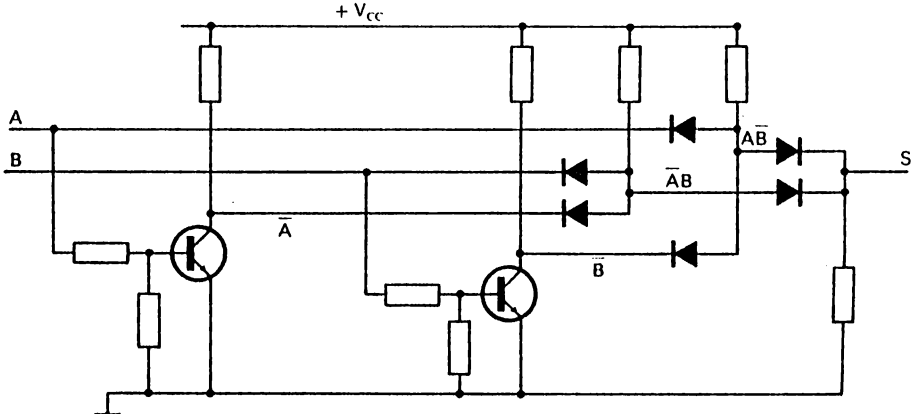


Fig. 1. 16

4.3 Le complément du OU-EXCLUSIF

En inversant la sortie d'une porte OU-EX (EXOR) on obtient la relation caractérisée par la table de vérité suivante :

A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

En examinant le tableau de Karnaugh n° 2 du paragraphe 4.2, nous voyons que cette relation peut s'exprimer sous la formule :

$$S = AB + \bar{A}\bar{B} \qquad (a)$$

Mais puisque nous l'obtenons par inversion de la porte EXOR, nous pouvons aussi écrire

$$S = \overline{A\bar{B} + \bar{A}B} \qquad (b)$$

Il y a donc équivalence des relations (*a*) et (*b*).

Appliquons donc le théorème de Morgan à la relation (*b*) :

$$\overline{A\bar{B} + \bar{A}B} = \overline{A\bar{B}} \cdot \overline{\bar{A}B},$$

avec $\overline{A\bar{B}} = \bar{A} + B$ et $\overline{\bar{A}B} = A + \bar{B}$ (de Morgan) d'où

$$\overline{A\bar{B} + \bar{A}B} = (\bar{A} + B) \cdot (A + \bar{B})$$
$$= \bar{A} \cdot (A + \bar{B}) + B \cdot (A + \bar{B})$$
$$= \bar{A}A + \bar{A}\bar{B} + BA + B\bar{B}$$

mais $A\bar{A}$ et $B\bar{B}$ sont nuls car il n'y a pas d'intersection entre A et $\bar{A}$ et entre B et $\bar{B}$ (A et $\bar{A}$ sont toujours opposés et donnent 0×1 ou 1×0) :

$$S = \bar{A}\bar{B} + AB \quad \text{ou} \quad AB + \bar{A}\bar{B}$$

4.4 Symbole de la relation OU-EX en algèbre binaire

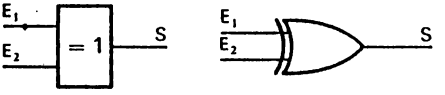
La relation OU-EX (EXOR) étant assez couramment utilisée en algèbre binaire, on a trouvé commode de créer un signe particulier pour l'exprimer, ce qui évite de développer chaque fois son expression en fonction de A et B.

Ce signe est $\oplus$.

On écrira donc $A \oplus B = A\bar{B} + \bar{A}B$

et $\overline{A \oplus B} = AB + \bar{A}\bar{B}$

Quant aux symboles graphiques de la porte OU-EX ou EXOR ils sont présentés en figure 1. 18.



a : OU-EX *b* : EXOR

Fig. 1. 18

5. TECHNOLOGIE ET CARACTÉRISTIQUES DES PORTES ÉLECTRONIQUES

Il est bon d'avoir des connaissances sur les technologies utilisées dans la conception des portes proposées par les constructeurs. Cela permet de les utiliser à bon escient et d'éviter quelques mésaventures.

5.1 Caractéristiques essentielles des circuits

Reprenons comme exemple le circuit en éléments discrets que nous avons proposé pour la porte ET (*fig.* 1. 19).

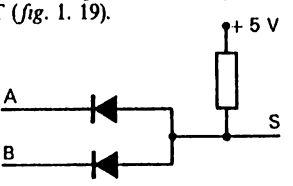


Fig. 1. 19

Supposons que A soit à « 1 » et B à « 0 ». Pour que le conducteur B reste à 0 volt, il faut que l'élément qui fournit la valeur B (en amont) soit capable d'*absorber* un courant. Si ce n'était pas le cas, la porte ne pourrait plus jouer son rôle car la tension s'élèverait sur le conducteur B.

La sortie S, elle-même, doit permettre

— si elle est à 1, de fournir le courant nécessaire à l'exploitation de l'information sans une trop forte chute de tension,

— si elle est à 0, d'absorber le courant provenant des circuits d'exploitation sans élévation notable de sa tension.

C'est pourquoi les constructeurs donnent très généralement les caractéristiques suivantes :

— Courant fourni (sortant) sur les « entrées » qui sont au niveau 0 (niveau *bas* appelé, en anglais, *low level* ou *L*).

— Courant absorbé (entrant) sur les entrées qui sont au niveau 1 (niveau *haut*, en anglais *high level* ou *H*).

— Nombre de portes de même type qu'il est possible d'alimenter par la sortie de la porte en question.

— Fourchette de tension permise autour du niveau bas (en général quelques dixièmes de volt).

— Fourchette de tension permise autour du niveau haut (souvent plusieurs volts).

— Vitesse de passage du niveau bas au niveau haut.

— Vitesse de passage du niveau haut au niveau bas.

En ce qui concerne les fourchettes de tolérance de tension, on peut comprendre que plus elles sont importantes, moins la porte est sensible aux « bruits » parasites qui peuvent apparaître sur ses entrées.

Quant aux vitesses de montée ou de descente elles seront utiles pour prévoir l'utilisation de ces portes dans le cas où les signaux varient très vite (à des fréquences élevées).

Une autre caractéristique est le temps de réponse de ces portes. Si nous examinons la figure 1. 20, nous constatons que la sortie réagit avec un certain retard aux conditions d'entrée, ce retard pouvant être différent pour chaque sens de variation du signal : ici le retard à la transition HL (du haut vers le bas), t_1, est plus faible que le retard t_2 à la transition LH (du bas vers le haut).

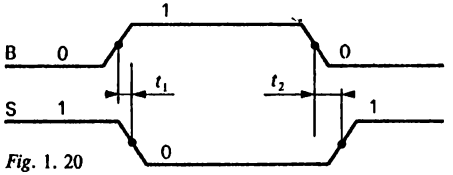


Fig. 1. 20

La moyenne $\frac{t_1 + t_2}{2}$ détermine le délai moyen de propagation pour la porte considérée.

Ce délai moyen peut varier de quelques nanosecondes à une centaine de nanosecondes.

Enfin il est important de connaître la puissance absorbée par ces circuits pour dimensionner l'alimentation en conséquence.

5.2 Les deux grandes familles de circuits intégrés

En attendant la découverte de nouvelles technologies, deux grandes familles de circuits intégrés sont proposées aujourd'hui sur le marché : les circuits TTL et les circuits CMOS.

5.2.1 Les circuits TTL

TTL signifie « Transistor-Transistor-Logic ». Ce sont des familles « tout transistors » (les diodes elles-mêmes sont constituées de transistors dont la base et le collecteur sont reliés.

Leurs caractéristiques sont les suivantes (pour les circuits dits « standards » de la série SN74) :

— Tension nominale d'alimentation : 5 V ± 0,5 V (ne jamais élever la tension au delà de 7 volts).

— Tension maximale d'entrée : 5,5 V.

— Température de fonctionnement :
entre 0 °C et 70 °C pour la série 74,
entre — 55 °C et 125 °C pour la série 54 (dite militaire).

— Tension maximum d'entrée pour un niveau bas (L) : 0,8 V.

— Tension minimum d'entrée pour un niveau haut (H) : 2 V.

Entre 0,8 volt et 2 volts la réponse en sortie sera incertaine et inexploitable.

— Tension maximum de sortie au niveau bas : 0,4 V.

— Tension minimum de sortie au niveau haut : 2,4 V.

— Courant d'entrée au niveau bas : — 1,6 mA.

— Courant d'entrée au niveau haut : 40 µA.

— Temps de propagation

t_1 (HL) nominal 7 ns
maximum 15 ns

t_2 (LH) nominal 11 ns
maximum 22 ns.

— Puissance moyenne absorbée de l'ordre de 10 mW par porte.

On trouvera en annexe un tableau des principaux circuits TTL signalés dans ce chapitre : amplis, inverseurs et portes à deux entrées.

Ces circuits sont groupés en deux catégories :

— la première comporte les portes à sortie simple, le câblage de cette sortie étant faite en général suivant le schéma du « totem-pole »,

— la seconde catégorie concerne les circuits dont la sortie est à « collecteur ouvert » : la charge de ce collecteur n'est pas comprise dans le circuit; elle doit donc être prévue à l'extérieur du circuit, entre l'alimentation et la sortie.

Nous verrons bientôt l'utilité de tels circuits. Il nous a paru, en attendant, pratique de regrouper sur un même tableau ces deux types de circuits.

5.2.2 Les circuits CMOS

Ces circuits, basés sur la technologie des MOS complémentaires se distinguent essentiellement par une consommation nettement plus faible (près de 100 fois!), de l'ordre de 0,1 mW par porte.

En contrepartie les temps de propagation sont plus longs, de l'ordre de 50 à 100 ns.

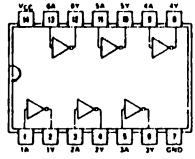
On notera aussi qu'ils exigent quelques précautions d'emploi, ces circuits pouvant être détruits lors de leur manipulation par des décharges électriques provenant de l'électricité statique. Aujourd'hui ils sont munis de protections d'entrée qui n'éliminent cependant pas tout risque.

Nous n'en dirons pas davantage sur ce sujet, renvoyant le lecteur à la documentation des constructeurs. La plupart des applications proposées dans cet ouvrage seront à base de circuits TTL.

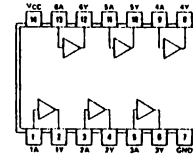
Annexe

– Série 74 –

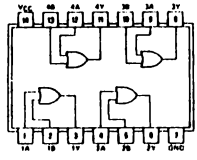
7404
6 inverseurs



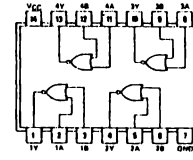
7407
6 amplis non inverseurs (drivers)



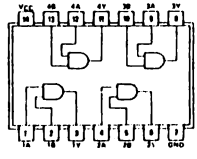
7432
4 portes OU à 2 entrées (OR)



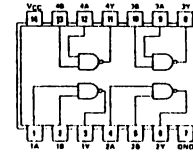
7402 et 7428
4 portes OU-NON à 2 entrées (NOR)



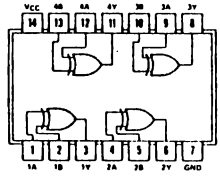
7408
4 portes ET à 2 entrées (AND)



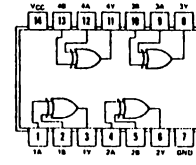
7400 et 7437
4 portes ET-NON à 2 entrées (NAND)



7486
4 portes OU EXCLUSIF (EXOR)

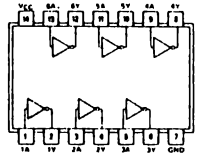


7436
4 portes OU-EXCLUSIF (EXOR)

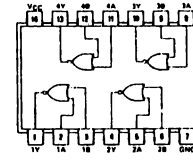


– Série 74 à collecteur ouvert –

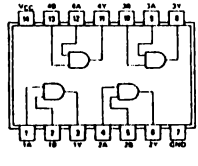
7405
6 inverseurs



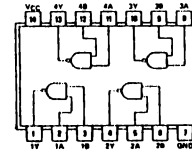
7433
4 portes OU-NON à 2 entrées (NOR)



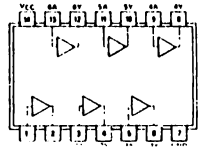
7409
4 portes ET à 2 entrées (AND)



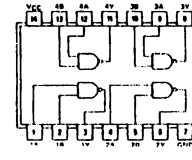
7401
4 portes ET-NON à 2 entrées (NAND)



7417
6 amplis non inverseurs (drivers)



7403
4 portes ET-NON à 2 entrées (NAND)



CHAPITRE 2

Les circuits logiques complexes

INTRODUCTION

La complexité d'un circuit logique peut être due à la quantité importante des informations d'entrée à traiter ou à la complexité même des lois d'association.

Deux outils essentiels sont alors utiles pour venir à bout de cette complexité : les tableaux de Karnaugh et l'algèbre binaire (ou algèbre de Boole).

1. LES TABLEAUX DE KARNAUGH

1.1 Principe de construction d'un tableau à plus de deux variables d'entrée

Les figures 2. 1 *a* et 2. 1 *b* montrent ce principe de construction : la figure 2. 1 *a* est l'application à trois variables d'entrée A, B, C.

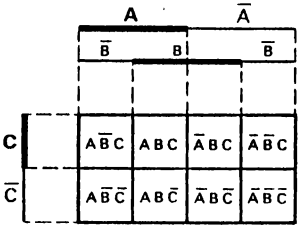


Fig. 2. 1 *a*

La figure 2. 1 *b* permet l'étude des combinaisons de cinq variables d'entrée.

Sur ce dernier tableau, la case marquée d'une croix correspond à l'expression $A \cdot B \cdot \bar{C} \cdot D \cdot E$ (A et B et $\bar{C}$ et D et E). Les quatre cases hachurées correspondent à $\bar{A} \cdot C \cdot \bar{D}$; on notera que dans ce cas, les termes B ou $\bar{B}$ et E ou $\bar{E}$ ne figurent pas, car la zone hachurée recouvre aussi bien B que $\bar{B}$ et E que $\bar{E}$.

On aurait pu présenter un tableau à 5 variables d'entrée tel que celui de la figure 2. 1 *b* en disposant 2 variables (A et B par exemple) en abscisse et trois variables en ordonnée (C, D et E).

Sur ce même principe on pourra étudier des combinaisons de très nombreuses variables.

En pratique il est rare que l'on ait à en étudier plus de huit ou dix en même temps.

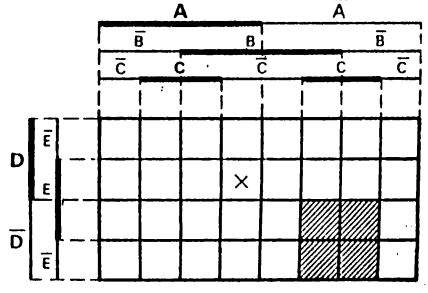


Fig. 2. 1 *b*

1.2 Caractéristiques liées à la construction du tableau

Si l'on examine les expressions correspondant à chaque case du tableau de la figure 2. 2, on constate que chacune ne diffère de sa voisine, située à côté sur l'horizontale ou la verticale, que par un seul terme.

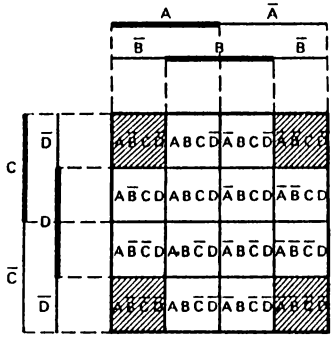


Fig. 2. 2.

Il en est de même pour la première et la dernière expression de chaque ligne, ou pour la première et la dernière expression de chaque colonne.

Il faut en fait, examiner tout tableau de Karnaugh comme s'il était inscrit sur un cylindre à axe vertical ou sur un cylindre à axe horizontal.

Sur cette même figure 2. 2, considérons les quatre cases des coins de ce tableau (marquées en grisé) : elles correspondent à l'expression

$$S = A\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D}$$

soit $S = (A + \bar{A})\bar{B}C\bar{D} + (A + \bar{A})\bar{B}\bar{C}\bar{D}$

comme $A + \bar{A} = 1$

$$S = \bar{B}C\bar{D} + \bar{B}\bar{C}\bar{D} = \bar{B}\bar{D}(C + \bar{C}) = \bar{B}\bar{D}$$

Il s'agit bien de quatre cases adjacentes, carrefour des zones $\bar{B}$ et $\bar{D}$ qui ont été artificiellement séparées sur le tableau de Karnaugh pour les besoins de sa construction.

1.3 Variante d'utilisation du tableau de Karnaugh

Soit la relation $S = A\bar{B}C + AB\bar{C}$.

Au lieu d'établir une table de vérité des huit cas possibles (deux valeurs de A × 2 valeurs de B × 2 valeurs de C), il est commode d'utiliser le tableau de Karnaugh en mettant, dans ses cases, des 1 pour les combinaisons qui correspondent à la relation S et des 0 dans les autres (voir *fig.* 2. 3). L'interprétation s'en fait de la manière suivante :

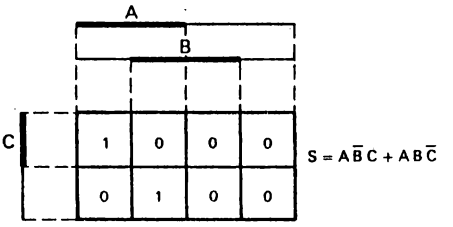


Fig. 2. 3

S est égale à 1
si A = 1, B = 0 et C = 1
ou si A = 1, B = 1 et C = 0
Dans tous les autres cas, S est égale à 0.

2. ALGÈBRE DE BOOLE

2.1 Rappels

Dans le chapitre précédent nous avons déjà vu les relations particulières suivantes :

$$A + \bar{A} = 1$$

$$A \cdot \bar{A} = 0$$

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

2.2 Distributivité de la multiplication vis-à-vis de l'addition

Le tableau de Karnaugh permet de vérifier la distributivité de l'opération ET vis-à-vis du OU, c'est-à-dire de la multiplication logique vis-à-vis de l'addition logique. Partant d'un tableau à quatre variables (la variable D n'est pas indispensable, elle figure pour montrer la généralité de cette loi), étudions la relation

$$S = A \cdot (B + C) \qquad (\textit{fig.}\ 2.\ 4)$$

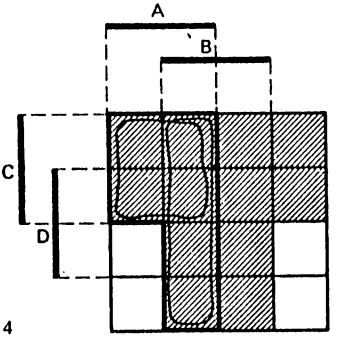


Fig. 2. 4

Le domaine de B + C correspond à toute la zone en grisé. A . (B + C) correspond à la zone entourée d'un trait gras.

Cette même zone correspond à la réunion des zones entourées d'un trait arrondi. Ces deux zones correspondent d'une part à A . C et d'autre part à A . B.

$$S = A \cdot B + A \cdot C$$

d'où

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

2.3 Distributivité de l'addition vis-à-vis de la multiplication

Plus surprenant est ce caractère de distributivité qui n'existe pas dans l'algèbre classique. Il s'exprime sous la formule

$$A + B \cdot C = (A + B) \cdot (A + C)$$

Sur la figure 2. 5 la zone A + B a été hachurée dans un sens et la zone A + C dans l'autre sens.

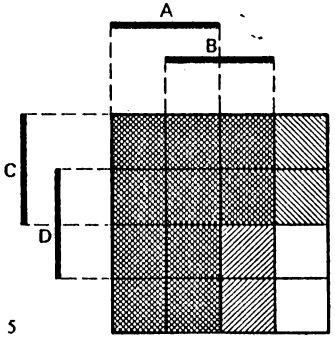


Fig. 2. 5

On voit que l'intersection de ces deux zones correspond bien à A + BC.

On utilisera aussi cette formule, dans un but de simplification, dans le sens contraire :

$$(A + B)(A + C) = A + BC$$

2.4 Autres identités remarquables

2.4.1 $S = A + \overline{A}B$

De l'identité précédente on peut déduire :

$$S = (A + \overline{A}) \cdot (A + B)$$

mais $A + \overline{A} = 1$
d'où

$$A + \overline{A}B = A + B$$

Notons que ceci peut se déduire aussi de la construction d'un tableau de Karnaugh (*fig.* 2. 6).

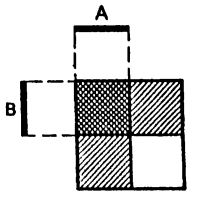


Fig. 2. 6

2.4.2 $S = AB + \overline{A}C$

Le tableau 2. 7 nous montre que la zone BC est incluse dans cette relation. Nous pouvons aussi rajouter le terme nul $A \cdot \overline{A}$

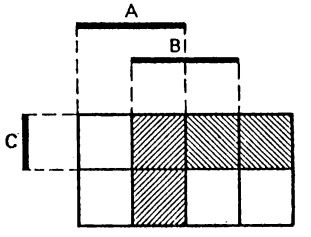


Fig. 2. 7

$$S = A \cdot B + B \cdot C + A \cdot \overline{A} + \overline{A} \cdot C$$
$$S = B(A + C) + \overline{A}(A + C)$$
$$S = (\overline{A} + B) \cdot (A + C)$$

d'où $A \cdot B + \overline{A} \cdot C = (\overline{A} + B) \cdot (A + C)$

2.5 Le « OU-EXCLUSIF » à trois variables

Le « ou-exclusif » à trois variables ABC est tel que S ne vaut 1 que si A = 1 avec B = 0 et C = 0 ou B = 1 avec A = 0 et C = 0 ou enfin C = 1 avec A = 0 et B = 0. Le tableau de Karnaugh correspondant est celui de la figure 2. 8. La relation « OU-EXCLUSIF » à trois variables est donc

$$S = A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C$$

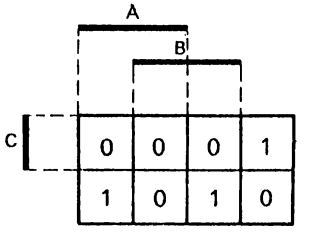


Fig. 2. 8

2.6 Combinaisons de « OU-EXCLUSIF »

Qu'en est-il alors de la relation

$$S = (A \oplus B) \oplus C$$

On peut utiliser l'algèbre de Boole pour expliciter cette expression :

$$\begin{aligned} S &= (A\overline{B} + \overline{A}B) \oplus C \\ &= (A\overline{B} + \overline{A}B) \cdot \overline{C} + \overline{(A\overline{B} + \overline{A}B)} \cdot C \\ &= A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + (\overline{A\overline{B}} \cdot \overline{\overline{A}B})C \\ &= A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + [(\overline{A} + B) \cdot (A + \overline{B})] \cdot C \\ &= A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + (\overline{A} \cdot \overline{B} + A \cdot B)C \end{aligned}$$

$$\boxed{S = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C + ABC}$$

On voit que cette relation diffère du OU EXCLUSIF à trois variables ABC par le dernier terme.

De plus, comme cette expression ne change pas si on remplace A par B, B par C et C par A on peut conclure que :

$$(A \oplus B) \oplus C = A \oplus B \oplus C = C \oplus A \oplus B \text{ etc.}$$

Avec un peu de soin on aurait pu construire directement le tableau de Karnaugh de cette relation (*fig.* 2. 9). S change de valeur chaque fois que l'une des variables change de valeur.

C'est le principe d'un va-et-vient électrique à trois interrupteurs.

En poursuivant, on pourrait mettre en évidence que la sortie (lampe) d'un interrupteur à quatre va-et-vient, peut s'exprimer sous la forme

$$S = A \oplus B \oplus C \oplus D$$

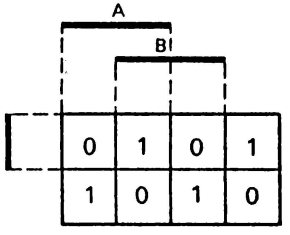


Fig. 2. 9

A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Fig. 2. 10

Il est intéressant d'observer la table de vérité de la relation $S = A \oplus B \oplus C$ (*fig.* 2. 10). On y remarque que S = 1 lorsque le nombre d'entrées de valeur 1 est impair.

S est appelé, pour cette raison, relation d'imparité. Un circuit permettant d'obtenir cette relation à partir de *n* variables d'entrée, est souvent utilisé en électronique digitale pour tester l'imparité (ou la parité) de l'ensemble des valeurs de ces variables.

3. RÉALISATION DES CIRCUITS LOGIQUES

A partir des outils essentiels que sont les tableaux de Karnaugh et l'algèbre de Boole, en s'aidant éventuellement des tables de vérité, il est possible de créer un circuit logique répondant à une relation donnée, à condition cependant...

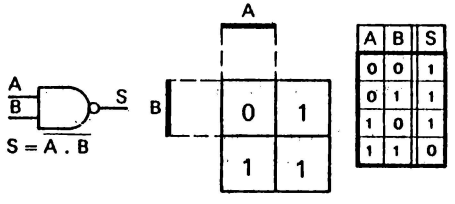
– de respecter les contraintes électroniques exigées des circuits employés,

– de prévoir ces circuits en fonction de la nature des éléments (portes) dont on dispose.

Il est en effet toujours possible de bâtir différents circuits, utilisant des portes différentes, pour répondre à un problème donné. Un exemple nous en est fourni par les multiples possibilités des portes NAND (ET NON) et NOR (OU-NON).

3.1 Diverses utilisations de la porte NAND

Nous avons vu que dans sa fonction de base, la porte NAND s'analyse par le tableau de Karnaugh et la table de vérité de la figure 2. 11.



A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Fig. 2. 11

3.1.1 Fonction « inverseur »

Si l'on connecte les deux entrées de la porte NAND à une même donnée A, la sortie donnera la valeur $\overline{A}$. Nous avons constitué ainsi un inverseur (*fig.* 2. 12).

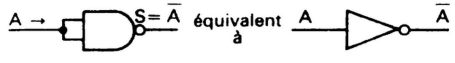


Fig. 2. 12

3.1.2 Fonction OR

La relation entre entrées et sortie, $S = \overline{\overline{A}\,\overline{B}}$, peut se traduire par

$$S = \overline{A} + \overline{B}$$

ce qui permet d'utiliser cette porte comme une porte OR de $\overline{A}$ et $\overline{B}$, ou de A et B pour peu que l'on inverse, avant l'entrée, les données A et B, par exemple par deux portes NAND (*fig.* 2. 13).

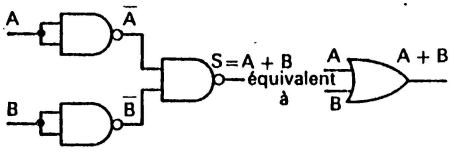


Fig. 2. 13

3.1.3 Fonction NOR

Au schéma de la figure 2. 13, il suffira d'ajouter une porte NAND câblée en inverseur, pour avoir une porte NOR.

3.1.4 Fonction AND

Sur le même principe, la porte AND nécessite l'emploi de deux portes NAND (*fig.* 2. 14).

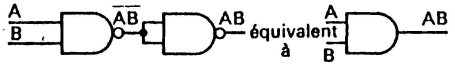


Fig. 2. 14

3.1.5 Fonction EX-OR

Elle est réalisée par le schéma de la figure 2. 15.

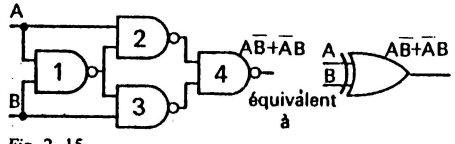


Fig. 2. 15

Elle correspond à la relation $S = A\overline{B} + \overline{A}B$. Pour que la porte NAND de sortie n° 4 nous fournisse cette valeur, il faut qu'avant inversion nous ayions :

$$\overline{A\overline{B} + \overline{A}B} = \overline{A\overline{B}} \,.\, \overline{\overline{A}B} = (\overline{A} + B)(A + \overline{B})$$

Les deux entrées de cette porte doivent donc être $\overline{A}+B$ (sortie de la porte n° 2) et $A+\overline{B}$ (sortie de la porte n° 3).

La porte n° 2 devant fournir $\overline{A} + B$, devrait donner avant inversion $\overline{\overline{A} + B} = A \,.\, \overline{B}$.

Ses deux entrées doivent correspondre en principe à A et $\overline{B}$.

De même les deux entrées de la porte n° 3 devraient être B et $\overline{A}$.

Il faudrait donc deux portes NAND supplémentaires, montées en inverseur pour fournir $\overline{A}$ et $\overline{B}$ à partir de A et B.

Le montage proposé supprime une porte NAND :

La porte n° 1 fournit $\overline{AB}$.

En entrée de la porte 2 nous avons donc A et $\overline{AB}$, et en sortie de cette même porte :

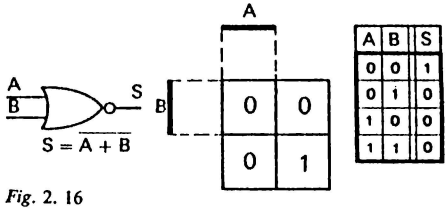
$$\overline{A \,.\, \overline{AB}} = \overline{A} + AB = \overline{A} + B \qquad \text{(voir § 2.4.1)}$$

ce que nous voulions obtenir.

La même démonstration montre que l'on a bien $A + \overline{B}$ en sortie de la porte 3.

3.2 Diverses utilisations de la porte NOR

Les caractéristiques de la porte NOR sont rappelées par le tableau de Karnaugh et la table de vérité de la figure 2. 16. Dans sa fonction de base, la porte donne $\overline{A + B}$.



A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Fig. 2. 16

3.2.1 Fonction «inverseur»

Comme pour la porte NAND, si l'on connecte à une même donnée d'entrée A les deux entrées de la porte NOR, on obtient un inverseur (*fig.* 2. 17).

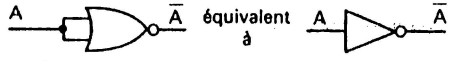


Fig. 2. 17

3.2.2 Fonction AND

La relation entre la sortie et les deux entrées $S = \overline{A + B}$ peut s'écrire sous la forme :

$$S = \overline{A} \,.\, \overline{B}$$

Si l'on inverse au préalable, par deux portes NOR, les deux variables d'entrée, on obtient une porte AND :

$$S = A \,.\, B \qquad \text{(cf. } fig.\text{ 2. 18).}$$

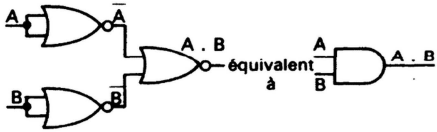


Fig. 2. 18

3.2.3 Fonction OR

En inversant la sortie d'une porte NOR, on obtient une porte OR (*fig.* 2. 19).

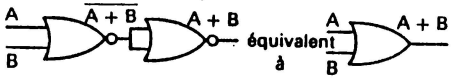


Fig. 2. 19

3.2.4 Fonction EX-OR

La dernière porte doit fournir $A\overline{B} + \overline{A}B$, c'est-à-dire, avant inversion :

$$\overline{A\overline{B} + \overline{A}B} = \overline{A\overline{B}} \cdot \overline{\overline{A}\overline{B}} = (\overline{A} + B)(A + \overline{B})$$
$$= \overline{A}A + \overline{A}\overline{B} + AB + B\overline{B} = AB + \overline{A}\overline{B}$$

Les deux entrées de cette porte doivent donc être AB et $\overline{A}\overline{B}$.

En se référant au § 3.2.2, on peut obtenir ces valeurs par quelques portes NOR, suivant le schéma 1 de la figure 2. 20.

On peut aussi utiliser le montage du schéma 2 de cette même figure.

3.3 Utilisation de portes à entrées multiples

Les fonctions AND, OR, NAND et NOR ne se limitent pas à deux entrées.

3.3.1 Pour réaliser la fonction

$$A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H$$

on pourra disposer des portes AND en cascade comme sur la figure 2. 21 *a* ou imaginer une porte AND à huit entrées (*fig.* 2. 21 *b*).

En fait, les constructeurs nous proposent soit une porte AND à quatre entrées, soit une porte NAND (donnant $\overline{ABCDEFGH}$, à huit entrées et dont il suffira d'inverser la sortie.

La porte AND à quatre entrées est, en technologie TTL la 7421 (deux portes), la porte NAND à huit entrées, la 7430 (une porte).

Dans cette même technologie, nous pouvons trouver : une porte AND à trois entrées : la 7411 (trois portes), une porte NAND à trois entrées : la 7410 (trois portes), une porte NAND à quatre

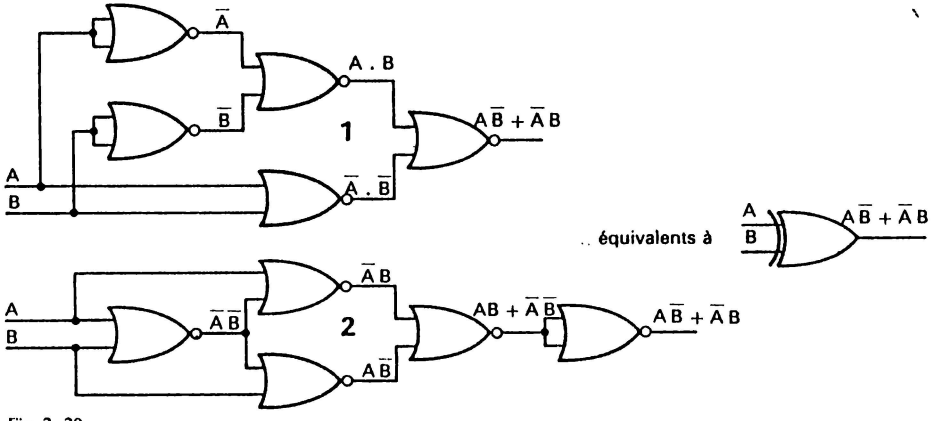


Fig. 2. 20

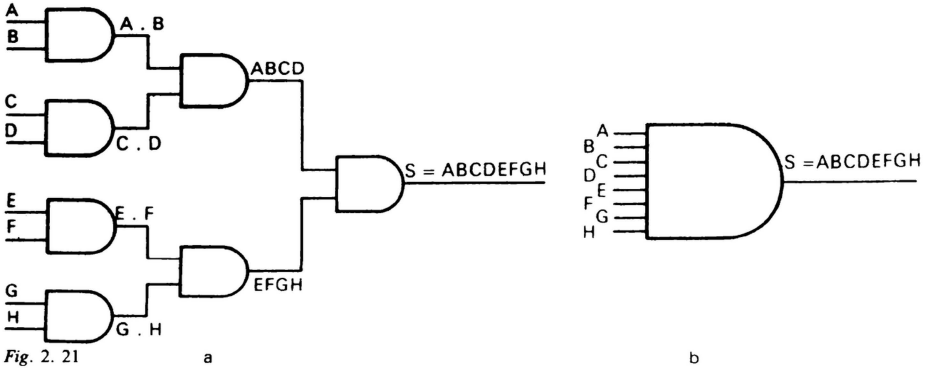


Fig. 2. 21 a b

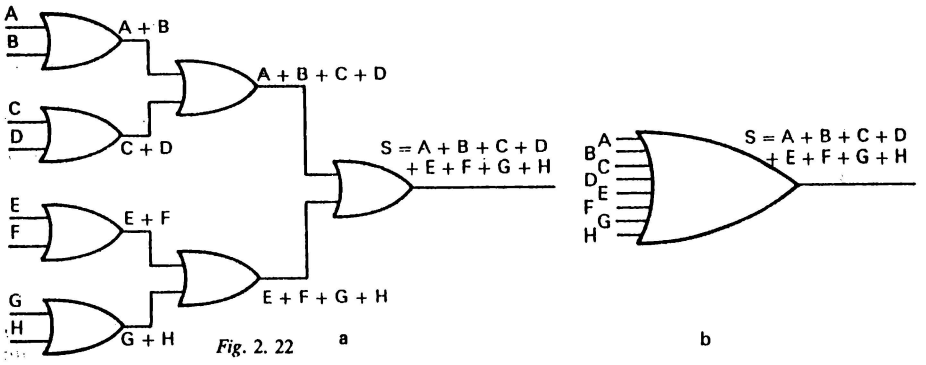


Fig. 2. 22 a b

entrées : 7420-7440 ou 74140 (deux portes), une porte NAND à huit entrées : 7430 (une porte), une porte NAND à treize entrées : 74133 (une porte).

On ne s'étonnera pas de trouver plus de variété en portes NAND qu'en portes AND, étant donnée la polyvalence de ces portes NAND.

En particulier n'oublions pas que $S = \overline{ABCD}$ est équivalent à $S = \overline{A} + \overline{B} + \overline{C} + \overline{D}$.

En effet, par simple application du théorème de Morgan :

$$S = \overline{A \cdot BCD} \qquad = \overline{A} + \overline{BCD}$$
$$= \overline{A} + \overline{B \cdot CD} \qquad = \overline{A} + \overline{B} + \overline{CD}$$
$$= \overline{A} + \overline{B} + \overline{C \cdot D} = \overline{A} + \overline{B} + \overline{C} + \overline{D}$$

3.3.2

Pour réaliser la fonction

$$A + B + C + D + E + F + G + H$$

on pourra ainsi disposer des portes OR en cascade ou imaginer une porte OR à huit entrées (*fig.* 2. 22 *a* et *b*).

Mais ici les constructeurs sont beaucoup moins prolixes, considérant que les portes NAND peuvent résoudre aussi élégamment les problèmes posés. Ils n'offrent donc que des portes OR ou NOR à deux entrées.

4. LES PORTES A « COLLECTEUR OUVERT »

Les constructeurs de portes en circuits intégrés TTL fournissent en plus des portes que nous venons de voir, les mêmes portes en « collecteur ouvert ».

4.1 Structure de la sortie des portes à « collecteur ouvert »

Cette sortie est présentée en figure 2. 23. Le collecteur du transistor de sortie est en l'air. Il ne peut fonctionner que si l'on dispose, à l'extérieur du circuit, une charge entre cette sortie collecteur et l'alimentation 5 V.

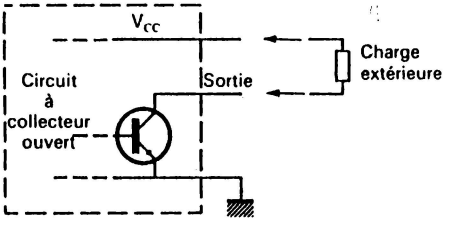


Fig. 2. 23

Cette charge étant branchée, on est ramené aux portes classiques, c'est-à-dire que la sortie, au niveau proche du 5 V (transistor bloqué) signifie 1 et, au niveau proche de 0 V (transistor saturé), signifie 0.

4.2 Branchement d'une diode électro-luminescente

Cette charge de collecteur peut être notamment une LED. Il faudra toutefois limiter le courant par l'adjonction d'une résistance de 500 Ω à 1 kΩ (*fig.* 2. 24).

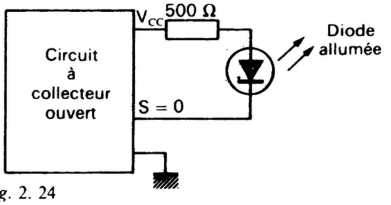


Fig. 2. 24

Si le niveau de sortie est 1, la diode sera éteinte (transistor bloqué).

Si le niveau de sortie est 0, la diode sera allumée.

Dans ce cas, la diode utilisée comme signal, s'allumant pour la valeur 0, donne donc l'inverse du signal de sortie du circuit.

Ce branchement d'une diode pourrait être utilisé sur un circuit classique (à collecteur fermé), à condition de le faire entre sortie et masse. Il présente cependant un inconvénient (*fig.* 2. 25).

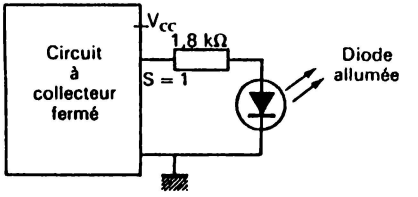


Fig. 2. 25

Le collecteur du circuit intégré ne peut débiter qu'un courant limité (environ 400 µA). Il faut donc mettre, en série avec la diode, une charge de 1,8 à 2 kΩ qui limite l'éclairement de cette diode.

Le signal fourni, n'inverse plus cette fois, le signal donné par le circuit.

4.3 Mise en parallèle des sorties à « collecteur ouvert »

L'intérêt essentiel de tels circuits réside en fait dans la possibilité de relier entre elles les sorties de plusieurs portes à « collecteur ouvert ».

Imaginons un système de surveillance nocturne d'un bâtiment par exemple : toutes les portes et fenêtres fermées fournissent des signaux ABC..., au niveau 1. Qu'un seul de ces signaux passe à 0, signalant l'ouverture d'une issue, le circuit de surveillance doit alors donner l'alerte.

Nous supposerons (*fig.* 2. 26) que les signaux aboutissent à quatre portes AND à trois entrées :

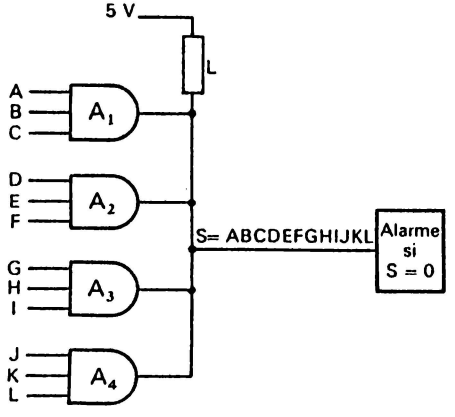


Fig. 2. 26

les portes A1, A2, A3 et A4 à collecteur ouvert. On peut alors relier toutes les sorties de ces portes entre elles et, à travers une seule charge L, au circuit d'alimentation.

En sortie de ce fil commun, nous aurons la fonction AND de toutes les entrées. On aura ainsi constitué ce que l'on appelle aussi un « ET fantôme ».

4.3.1 Calcul de la charge

Imaginons qu'une des sorties de A1, A2, A3 ou A4 soit au niveau bas. Elle absorbe alors un courant (à travers le transistor de sortie) dont la valeur varie suivant les types. Le courant du circuit standard est de 16 mA (il est noté I_{OL} dans les catalogues).

Le niveau 0 doit être caractérisé par un niveau de tension maximum de 0,4 volts.

La résistance de charge ne doit pas fournir plus que ce courant, d'où :

$$L_{min} = \frac{5\ V - 0{,}4\ V}{0{,}016\ A} \approx 300\ \Omega$$

et ceci à condition que le système d'alarme lui-même ne soit pas fournisseur de courant.

En revanche, si toutes les portes A fournissent 1 en sortie, il doit exister un léger courant entrant dans ces portes par leur sortie si l'on veut maintenir une tension significative du niveau haut (2,4 V mini). Ce courant est de l'ordre de 250 µA. Il faut de plus pouvoir alimenter l'aval, c'est-à-dire le système d'alarme. Supposons qu'il nécessite, pour le maintien du niveau haut une tension de 2,4 V et un courant de 300 µA. On peut alors calculer la valeur maximum de L

$$L_{max} = \frac{5 - 2{,}4}{4 \times 0{,}000\,25 + 0{,}000\,3} = \frac{2{,}6}{0{,}001\,3} = 2\,k\Omega$$

5. AIGUILLAGES DE DONNÉES

5.1 Un interrupteur inverseur logique

Grâce à quelques portes (*fig.* 2. 27 *a*) il est possible de construire un circuit analogue au circuit électrique de la figure 2. 27 *b*, la commande d'inversion se faisant par le contact C.

Dans les deux cas, lorsque l'interrupteur C est ouvert le signal 1 apparaît sur la sortie S2.

Lorsque l'interrupteur C est fermé, ce signal apparaît sur la sortie S1.

La donnée D, créée par l'interrupteur C agit ici pour orienter la donnée A sur l'un des deux fils de sortie (*fig.* 2. 27 *b*). A et D sont des données

binaires, mais ce montage leur donne des fonctions ou des rôles différents : on peut considérer A comme une information (donnée) d'entrée, alors que D aurait plutôt une fonction de commande.

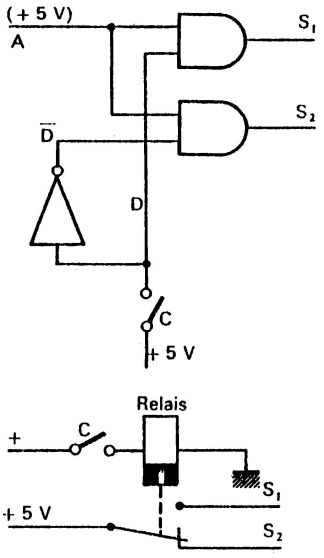


Fig. 2. 27

5.2 Inhibition

Il est même possible de créer une donnée ayant une nouvelle fonction : celle d'inhibition. Il suffit de disposer cette troisième donnée comme sur la figure 2. 28. Tant que l'interrupteur I n'est pas fermé, aucune sortie (ni S1 ni S2) ne peut transmettre la donnée A. Si I est fermé (J = 1), on est ramené au schéma précédent.

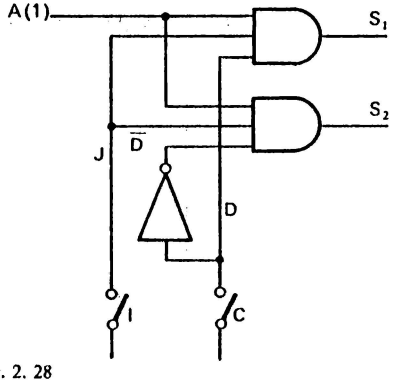


Fig. 2. 28

5.3 Aiguillages en parallèle

En mettant en parallèle plusieurs montages des figures 2. 27 *a* ou 2. 28, on peut obtenir l'équivalent de plusieurs interrupteurs inverseurs commandés par le même électro-aimant (*fig.* 2. 29).

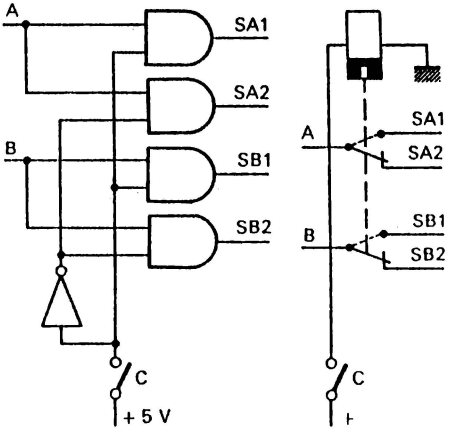


Fig. 2. 29

A noter que, dans ce cas, toutes les données entrant dans les circuits en parallèle sont orientés en même temps vers les sorties 1 ou (exclusif) vers les sorties 2.

5.4 Réseau d'aiguillages

En associant alors en cascade de tels circuits on peut constituer un réseau d'aiguillages analogue à celui d'une gare de triage.

Sur la figure 2. 30 on a symbolisé ce réseau par des interrupteurs inverseurs pour simplifier le schéma, et surtout le rendre plus « parlant » mais il faut imaginer chacun de ces interrupteurs construit suivant le schéma 2.27 *a* et mis en parallèle suivant le principe de la figure 2. 29.

On voit que l'établissement des valeurs de commande, (C1, C2, C3, C4, C5) permet d'obtenir l'aiguillage de la donnée A vers l'une des 32 sorties numérotées de 0 à 31. Ici, lorsque $C_5 = 1$, $C_4 = 0$, $C_3 = 1$, $C_2 = 0$ et $C_1 = 1$ on constate, en suivant le circuit matérialisé par un trait ininterrompu depuis l'entrée A jusqu'à la sortie, que la donnée va apparaître sur la sortie 21.

Que faudrait-il alors pour que ce soit la sortie 15 qui soit activée?

En partant de cette sortie 15, on voit qu'il faut $C_1 = 1$, puis $C_2 = 1$, puis $C_3 = 1$, puis $C_4 = 1$ enfin $C_5 = 0$. Une étude plus générale nous donnera la table de vérité suivante :

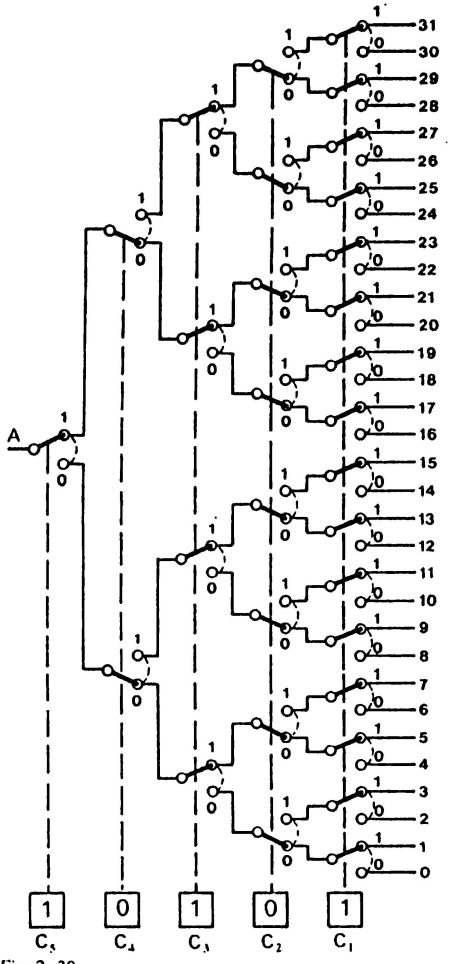


Fig. 2. 30

C_5	C_4	C_3	C_2	C_1	S
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
0	1	0	0	0	8
0	1	0	0	1	9
0	1	0	1	0	10
0	1	0	1	1	11
0	1	1	0	0	12
0	1	1	0	1	13
0	1	1	1	0	14
0	1	1	1	1	15
1	0	0	0	0	16
1	0	0	0	1	17
1	0	0	1	0	18
1	0	0	1	1	19
1	0	1	0	0	20
1	0	1	0	1	21
1	0	1	1	0	22
1	0	1	1	1	23
1	1	0	0	0	24
1	1	0	0	1	25
1	1	0	1	0	26
1	1	0	1	1	27
1	1	1	0	0	28
1	1	1	0	1	29
1	1	1	1	0	30
1	1	1	1	1	31

6. SYSTÈME BINAIRE PUR

Cette table de vérité permet de mettre en correspondance un groupe de valeurs binaires (C_5, C_4, C_3, C_2, C_1) autrement dit un nombre binaire, avec un nombre décimal.

En observant bien cette table de vérité, on peut trouver une loi générale d'établissement des relations entre le binaire pur et le décimal :

tous les nombres pour lesquels $C_5 = 0$ sont compris entre 0 et 15.

Dès que C_5 est à 1 les nombres décimaux sont $\geqslant 16$.

Dès que C_4 est à 1, les nombres décimaux sont $\geqslant 8$. Le nombre 10000 vaut 16 et le nombre 01000 (ou 1000) vaut 8. Le nombre décimal 24 (16 + 8) s'écrit 11000 en binaire, c'est-à-dire 10000 + 1000 (16 + 8).

6.1 Conversion du binaire en décimal

De façon plus générale, on constate que le 1 de la première colonne (C_1) vaut 1 en décimal, que le 1 de la colonne (C_2) vaut 2, celui de C_3 vaut 4, le 1 de C_4 vaut 8 et le 1 de C_5 vaut 16.

Si le nombre comportait n colonnes, le 1 de la colonne n vaudrait, en décimal, 2^{n-1}. Par exemple :

$$101 \text{ vaut } 2^2 + 2^0 = 4 + 1 = 5$$

(se rappeler que X^0 est toujours égal à 1)

$$1101010 = 2^6 + 2^5 + 2^3 + 2^1$$
$$= 64 + 32 + 8 + 2 = 106$$

6.2 Conversion du décimal en binaire

6.2.1 Par soustractions successives

Soit le nombre 87. On peut calculer quelle est la plus grande puissance de 2 qu'il contient. $64 < 87 < 128$. On peut donc ôter 64 de 87 :

$$87 - 64 = 23$$

23 contient 16 : $23 - 16 = 7$

7 contient 4 : $7 - 4 = 3$

3 contient 2 : $3 - 2 = 1$.

Donc $87 = 64 + 16 + 4 + 2 + 1$

$= 2^6 + 2^4 + 2^2 + 2^1 + 2^0$

Ce qui s'écrira en binaire pur :

1010111

6.2.2 Conversion par dichotomies successives

Le moyen de conversion précédent est un peu lourd. Un moyen plus élégant consiste à faire des dichotomies, c'est-à-dire à couper en deux ce nombre jusqu'à ce que l'on aboutisse à 1 — mais chaque fois que le nombre à couper est impair, on lui retire 1 que l'on place en mémoire (pour mémoire aussi on retire 0 des valeurs paires) :

Exemple :

87 impair	− 1	= 86 ,	86/2 = 43
43 impair	− 1	= 42 ,	42/2 = 21
21 impair	− 1	= 20 ,	20/2 = 10
10 pair	− 0	= 10 ,	10/2 = 5
5 impair	− 1	= 4 ,	4/2 = 2
2 pair	− 0	= 2 ,	2/2 = 1
1 impair	− 1	= 0	

Si on lit *de bas en haut* les valeurs mises en mémoire, on retrouve le nombre binaire 1010111.

Ceci n'est pas un procédé « magique »! On peut très bien l'expliquer et constater qu'il procède du même principe que celui qui nous a permis, au § 5.4 d'établir la correspondance binaire du nombre 15 en partant du réseau d'aiguillages de la figure 2. 30.

Nous aurons l'occasion, dans les chapitres suivants, de revenir largement sur ce système de numération en binaire pur appelé souvent code binaire pur.

7 SIMPLIFICATION DU RÉSEAU D'AIGUILLAGES

Si l'on transcrit le schéma de la figure 2. 30 en schéma électronique, doté de toutes ses portes AND, chaque interrupteur-inverseur nécessitant deux de ces portes, il faudra faire appel à 62 portes AND, ou 16 circuits intégrés 7408 par exemple.

Maintenant que nous avons analysé le code d'aiguillage, il est possible de modifier ce schéma par application directe du code sur la porte de sortie correspondante.

Ainsi, la sortie 15 correspondant au code 01111, nous construirons la porte 15 grâce à un circuit AND à 6 entrées correspondant à A (la donnée) et $\overline{C}_5 C_4 C_3 C_2 C_1$ (voir *fig.* 2. 31).

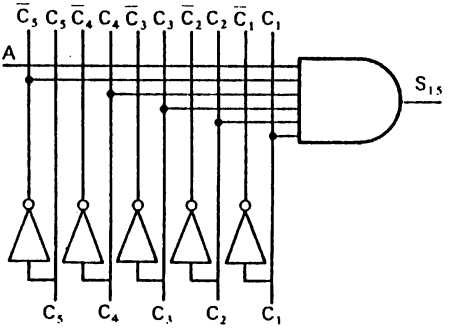


Fig. 2. 31 (Doc. Texas Instruments)

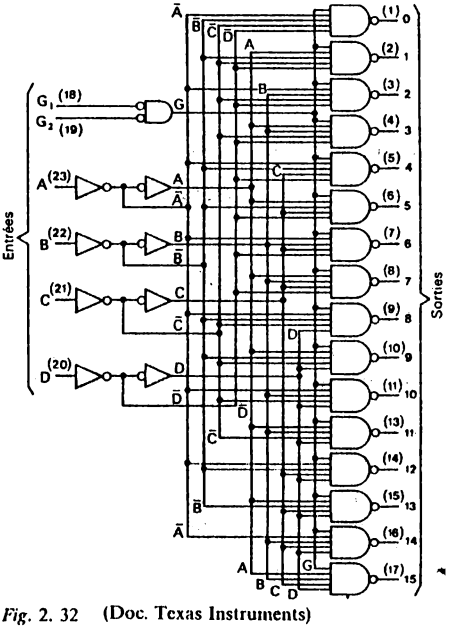


Fig. 2. 32 (Doc. Texas Instruments)

Il sera encore plus simple de faire appel à des circuits intégrés tels que les circuits 74154 qui permettent de disposer de 16 sorties à partir de quatre valeurs de commande (notées ici A pour C_1, B pour C_2, C pour C_3 et D pour C_4 (voir *fig.* 2. 32).

On pourra coupler deux de ces circuits, car ils disposent de deux entrées supplémentaires, G1 et G2, dont l'une pourra être utilisée pour la donnée à aiguiller et l'autre pour la *sélection* de l'un ou l'autre des circuits.

On notera aussi que les sorties étant des NAND, la valeur est inversée par rapport aux montages précédents.

Ces circuits sont appelés « décodeurs », car ils permettent de décoder un nombre binaire pur sur une sortie qui peut, par exemple, afficher le nombre décimal correspondant. Ils sont aussi appelés « démultiplexeurs », ce que nous expliquerons dans le chapitre 5.

EXERCICES

1. Simplifier l'expression suivante :

$$ABCD + B\bar{C}D\bar{E} + \bar{A}BCD + B\bar{C}DE$$

2. Un nombre binaire, ou un groupe de valeurs binaires se présentant sous la forme ABCD, on veut tester son imparité :

S, résultat du test, vaudra 1 si les valeurs 1 de ce groupe sont en nombre impair; S vaudra 0 si elles sont en nombre pair.

Par exemple, si A = 1, B = 0, C = 1 et D = 1, il y a trois valeurs égales à 1. 3 étant un nombre impair, S doit valoir 1.

a) Dresser le tableau de Karnaugh de la relation $S = f(A, B, C, D)$.

b) Exprimer cette relation par une formule d'algèbre de Boole, sous deux formes différentes.

c) En déduire deux schémas de circuits logiques permettant d'obtenir S à partir de A, B, C et D.

3. Une association est constituée de quatre membres dont un président.

Face à un problème épineux ils décident de voter à bulletin secret. Étant donné leur nombre il est décidé que le président (A) aura seul droit à un vote de poids double, les autres membres (B, C et D) intervenant pour une seule voix chacun.

L'un des membres étant électronicien leur propose alors une petite machine à voter comportant, en entrée, quatre petits boîtiers à boutons-poussoirs OUI (1) et NON (0) que les intéressés mettront sur leurs genoux, et en sortie une lampe qui devra s'allumer si la majorité de OUI est obtenue.

Imaginez une telle machine à voter en établissant auparavant le tableau de Karnaugh de la relation.

CHAPITRE 3

Les bascules

INTRODUCTION

Traiter des informations à travers des circuits logiques suppose que les informations qui se présentent à l'entrée ne sont pas connues à l'avance, sinon le résultat lui-même serait connu et il n'y aurait pas lieu de construire un circuit logique. Ces entrées vont être donc soumises, suivant les périodes, à des niveaux haut ou bas, c'est-à-dire à des impulsions plus ou moins longues.

L'électronique digitale est une électronique d'impulsions. On ne sera pas étonné de voir que les organes qui changent d'état (bascules) suivant les conditions d'entrée ou suivant le temps, y jouent un rôle important.

1. LA BASCULE BISTABLE R . S

1.1 Deux inverseurs bouclés

Une première idée de ce que peut être une bascule nous est donnée par le montage à deux inverseurs en boucle de la figure 3. 1.

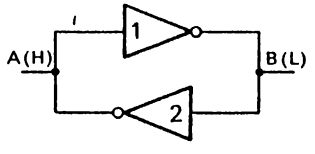


Fig. 3. 1

Lorsque ce montage est alimenté, un état d'équilibre se réalise très vite : par exemple A au niveau haut, B au niveau bas, et cet état est stable.

Malheureusement cet état est aussi imprévisible. Si l'on coupe et que l'on rétablisse l'alimentation à plusieurs reprises on peut obtenir l'un ou l'autre des deux états possible, au hasard.

Pour obtenir une configuration voulue, il faut «forcer» l'une des branches A ou B à O par exemple. En mettant A à la masse (*fig.* 3. 2) on aura 1 sur B, et cette situation restera stable même lorsqu'on relâchera l'interrupteur I_A.

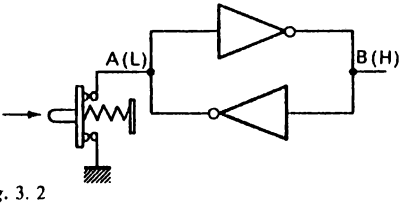


Fig. 3. 2

1.2 L'équivalent NOR

En remplaçant les inverseurs par des portes NOR, on obtient le schéma de la figure 3. 3 *a*, et la table de vérité de la figure 3. 3 *b*.

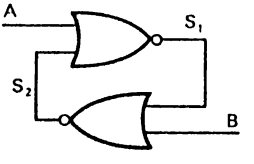


Fig. 3. 3 *a*

Observez bien ce qui se passe.

• A = 0 et B = 0 : la situation est indéterminée et s'établit au hasard comme précédemment. X et $\overline{X}$ signifient 0 et 1 ou 1 et 0.

• A = 0 et B = 1 : le niveau 1 sur B force la sortie S_1 à 1.

• A = 1 et B = 0 : le niveau 1 sur A force la sortie S_2 à 1.

• A = 1 et B = 1 : S_1 et S_2 passent au niveau bas (0).

A	B	S_1	S_2
0	0	X	$\overline{X}$
0	1	1	0
1	0	0	1
1	1	?	?

Fig. 3. 3 b

Toutefois cette dernière situation présente deux inconvénients :

a) les valeurs des sorties S_1 et S_2 ne sont plus complémentaires, ce qui est fâcheux pour une « bascule » qui est censée fournir en permanence deux valeurs complémentaires;

b) si, l'impulsion passée, A et B reviennent simultanément au niveau bas, les valeurs sur les sorties seront indéterminées (première situation).

Pour ces deux raisons, on cherchera à éviter le cas où A et B sont tous deux à 1.

La terminologie et les schémas habituels décrivant la bascule à double porte NOR sont un peu différents de ce que nous avons présenté ci-dessous :

Les sorties sont traditionnellement notées Q et $\overline{Q}$, ce qui sous-entend bien qu'elles sont complémentaires.

Les entrées sont nommées SET (mise à 1) et RESET (mise à 0), la commande SET devant forcer Q à 1 et la commande RESET à 0.

Le schéma lui-même est présenté sous la forme de la figure 3. 4. Il s'agit bien du même circuit.

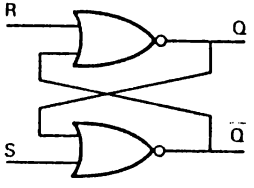


Fig. 3. 4

La commande SET devant mettre Q à 1, il s'agit bien du fil B (cas n° 2 où A = 0 et B = 1).

La commande RESET est sur le fil A.

On voit pourquoi cette bascule s'appelle bascule RS.

1.3 L'équivalent NAND

Sur le même principe il est facile de concevoir une bascule composée de deux portes NAND en boucle.

Les figures 3. 5 *a* et 3. 5 *b* en donnent le schéma et la table de vérité. La même analyse que précédemment nous montre cette fois que :

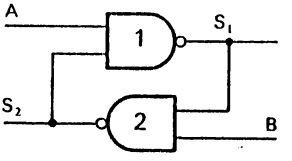
A = 1, B = 1 donne deux sorties complémentaires au hasard,

A = 1, B = 0 met la sortie S_2 à 1 (Reset),

A = 0, B = 1 met la sortie S_1 à 1 (Set),

A = 0, B = 0 est à éviter (sorties non complémentaires).

a



b

A	B	S_1	S_2
0	0	?	?
0	1	1	0
1	0	0	1
1	1	X	$\overline{X}$

Fig. 3. 5

Il faudra donc, dans la pratique, mettre les deux entrées A et B au potentiel +. Dans ce cas on ne peut plus envoyer d'impulsion positive sur ces fils de commande. On devra donc remplacer l'impulsion SET sur B en une impulsion négative, appelée donc $\overline{SET}$ sur A. De même B servira de commande $\overline{RESET}$.

Traduit dans les schémas et la terminologie habituelle ceci donne la figure 3. 6.

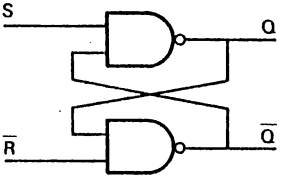


Fig. 3. 6

1.4 Remarque sur la polarisation des circuits TTL

Les circuits TTL sont conçus de telle sorte que les entrées laissées en l'air (non branchées) peuvent être considérées la plupart du temps comme portées au potentiel + 5 V. En effet ces sorties sont capables de débiter vers l'extérieur un faible courant de quelques micro-ampères. Pourtant il est préférable de relier les entrées au pôle + de l'alimentation à travers une résistance qui peut atteindre 30 kΩ (ou 30 × N kΩ pour N entrées reliées).

Si l'on veut qu'une des entrées soit en permanence à 0, il est impératif de la relier à la masse, soit directement, soit à travers une faible résistance.

1.5 Remarques sur la présentation des circuits en catalogue

La bascule composée de portes NAND est fréquemment incluse dans les circuits intégrés. On vient de voir qu'elle nécessite des commandes par impulsions négatives, l'état normal des commandes inactives étant alors 1. Par convention on note alors $\overline{S}$ et $\overline{R}$ (ou $\overline{PRESET}$ et $\overline{CLEAR}$) ces commandes.

Le schéma d'une bascule RS à NAND sera donc présenté par un constructeur sous les formes de la figure 3. 7. On y voit essentiellement que les bornes 1 et 2 aboutissent à des entrées du circuit précédées d'un petit rond, symbole d'inversion. Ce petit rond est le seul élément qui permette de savoir que l'impulsion de commande doit être un créneau négatif.

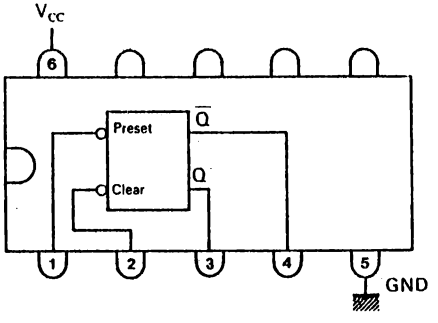


Fig. 3. 7

On pourra toujours rétablir la commande par signal positif en utilisant, entre cette commande et le circuit, le service d'un inverseur supplémentaire.

1.6 Perfectionnement de la bascule bistable, la bascule D

Les bascules que nous venons d'étudier se caractérisent par deux états stables possibles; c'est pourquoi elles s'appellent bistables. Nous avons constaté aussi la possibilité d'un troisième état où les deux sorties ne sont pas complémentaires et qui est à éviter. La solution est simple : il suffit de rendre complémentaires les commandes R et S.

Pourtant la solution représentée sur la figure 3. 8 n'apparaît pas très opérationnelle : Q et $\overline{Q}$ se contentent de fournir les valeurs D et $\overline{D}$, si bien que la bascule ne se justifie même pas.

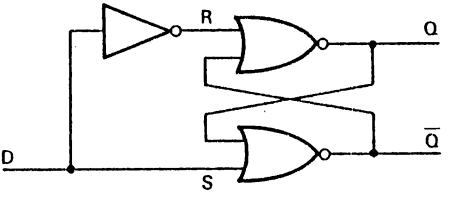


Fig. 3. 8

Un petit complément (*fig.* 3. 9) va donner tout son intérêt à cette bascule.

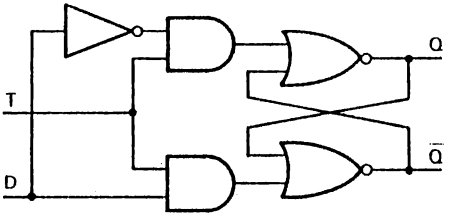


Fig. 3. 9

Il consiste à ajouter une commande sur laquelle parviendra un signal, sous forme d'un bref créneau, au moment où la valeur d'entrée sur D doit être enregistrée (apparaître sur la sortie Q).

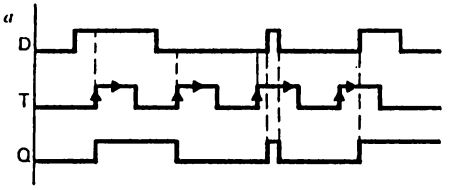
Pour la distinguer des autres commandes, cette commande sera appelée soit TRIGGER (déclencheur) soit encore CLOCK (horloge), et sera notée T ou Clk ou encore Cp (Clock pulse).

Dès que T = 1, la bascule transfère en Q la valeur présente sur D. Lorsque T retombe à 0, cette valeur de Q reste stable tant que S ou R ne sont pas activés.

1.7 Chronogramme et table de vérité de la bascule D

A défaut d'un oscilloscope à quatre traces nous permettant de suivre le comportement de la bascule suivant les niveaux apparaissant sur D et sur T, il est toujours possible de tracer ces variations sur un tableau où le temps est représenté en abscisse. On obtient alors la figure 3. 10 *a*.

a



b

D	T	Q	$\overline{Q}$
0	0	Q_{t-1}	$\overline{Q}_{t-1}$
1	0	Q_{t-1}	$\overline{Q}_{t-1}$
0	1	0	1
1	1	1	0

Fig. 3. 10

La table de vérité de la figure 3. 10 *b* utilise quelques expressions nouvelles : Q_{t-1} et $\overline{Q}_{t-1}$ indiquent que les sorties, lorsque T prend la valeur 0, conservent les niveaux qu'elles avaient précédemment. On pourra trouver aussi Q_{n-1} et $\overline{Q}_{n-1}$ ou Q_0 et $\overline{Q}_0$.

Cette bascule, reproduisant sur la sortie Q la valeur ou les différentes valeurs de D pendant toute la durée du créneau positif de T, et mémorisant la dernière valeur de D présente au moment du passage au niveau bas de T, s'appelle bascule D à « verrouillage » (en anglais bistable latch). Elle se trouve en quatre exemplaires, dans les circuits intégrés 7475 et 7477.

1.8 La bascule D du circuit intégré 7474

1.8.1 Schéma de la bascule

On peut construire la même bascule que celle de la figure 3. 9 en n'utilisant que des portes NAND. Une bonne précaution consiste à « conformer » la donnée à son arrivée, pour parer au risque qu'elle ne soit trop atténuée. On obtient alors la figure 3.11.

La conception de la bascule intégrée 7474 (deux par boîtier) hérite directement de ce schéma. L'utilisation systématique de portes NAND à trois entrées permet de rajouter les fonctions Set et Reset appelées ici PRESET et CLEAR, c'est-à-dire, rappelons-le, armement (Q à 1) et mise à zéro (Q à 0). Ce schéma est représenté en figure 3. 12.

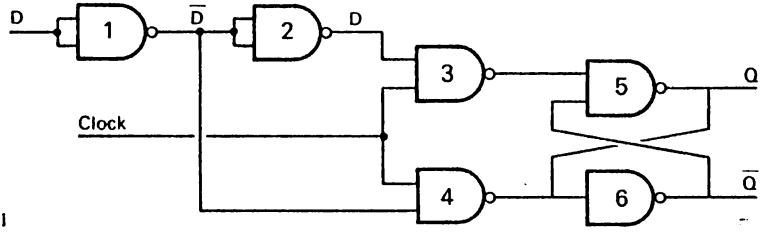


Fig. 3. 11

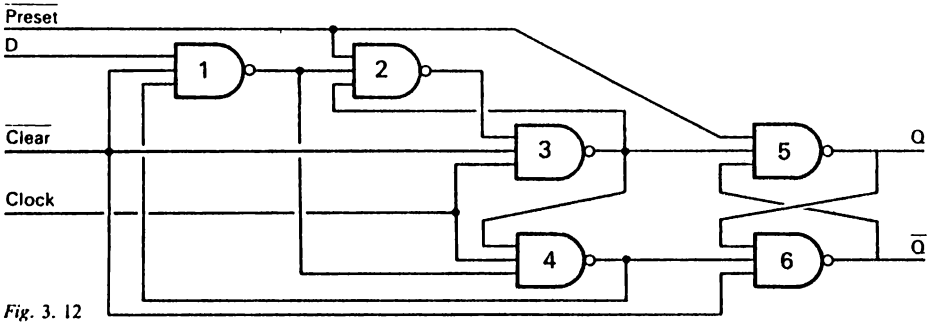


Fig. 3. 12

Comme les commandes Preset et Clear agissent par une impulsion négative, elles figurent sur ce schéma par l'indication complémentaire Preset et Clear, mais les catalogues ne le spécifient que par le petit rond du schéma simplifié et par la table de vérité qui explicite le fonctionnement (*fig.* 3. 13).

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$

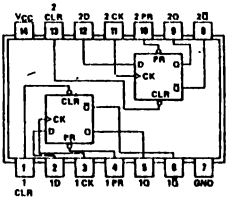


Fig. 3. 13 Double bascule D-7474. (Doc. Texas Instruments)

1.8.2 Fonctionnement

La grande caractéristique de cette bascule réside dans le fait qu'elle ne charge et ne conserve que la valeur de D *présente au moment du front montant de l'horloge.*

Pour comprendre ce fonctionnement, observez bien la figure 3. 14 qui montre le chronogramme des situations possibles :

A l'instant 1, D est au niveau bas. Le front montant de l'horloge enregistre cette valeur. Q reste à 0 jusqu'au prochain front montant de l'horloge, et malgré les variations qu'enregistre D un peu plus tard.

A l'instant 2, le deuxième front montant de l'horloge a lieu alors que D est à 1. Q passe à 1 et y restera jusqu'au prochain front montant de l'horloge, malgré les variations ultérieures de D.

On peut comprendre ce fonctionnement en analysant le schéma de la figure 3. 12 et notamment en étudiant le rôle de la liaison existant entre la sortie de la porte 4 et l'entrée de la porte 1. On verra alors que si, à l'origine, D = 0 et Clk = 1, la sortie de la porte 4 est au niveau bas et inhibe la porte 1 dont la sortie restera bloquée à 1 pendant tout le temps où Clk est à 1.

Lorsque Clk retombe à 0, ce sont alors les portes 5 et 6 qui gardent les valeurs de Q et $\bar{Q}$ comme nous l'avons vu précédemment.

Une telle bascule, qui n'est actionnée qu'au moment d'un front montant de l'horloge, s'appelle en anglais «edge triggered», tout comme celles qui seraient actionnées par un front descendant.

1.8.3 Applications

L'intérêt d'une telle bascule réside en grande partie dans sa possibilité de mémoriser un état passager. Nous en avons vu une application dans le chapitre 1. En généralisant cette application, imaginons que le fil D soit branché au + 5 V, et que le fil d'horloge passe par un rupteur (contact de porte, touche de contact, électro-transistor) qui coupe son alimentation ou au contraire envoie un créneau positif de tension. (Il suffira de mettre à l'entrée de la bascule un inverseur si nécessaire pour transformer toutes ces actions en créneau positif.) La bascule étant au préalable mise à zéro, ce créneau déclenchera le passage à 1 de Q, de façon permanente, *même si l'action cesse*, tant que Clear n'aura pas été actionné.

Le niveau 1 de Q pourra à son tour être concrétisé, à travers un amplificateur, par un signal sonore ou lumineux.

Dans le domaine du traitement de l'information, les bascules D à verrouillage ou à commande par front ont de larges applications que nous verrons plus loin.

1.9 Les bascules « maître-esclave »

1.9.1 La bascule T

En reprenant le schéma de la bascule RS avec commande d'horloge tel que celui de la figure 3. 15 *a*, imaginons comment on pourrait obtenir le basculement des valeurs Q et $\bar{Q}$ à chaque impulsion d'horloge.

Pour avoir Q = 0 il faudrait que l'impulsion d'horloge agisse sur la porte 1 (Reset), et pour avoir Q = 1, il faudrait qu'elle agisse sur la porte 2 (Set).

Imaginons donc de relier Q à R et $\bar{Q}$ à S (*fig.* 3. 15 *b*) et optons pour une situation de départ où Q = 1.

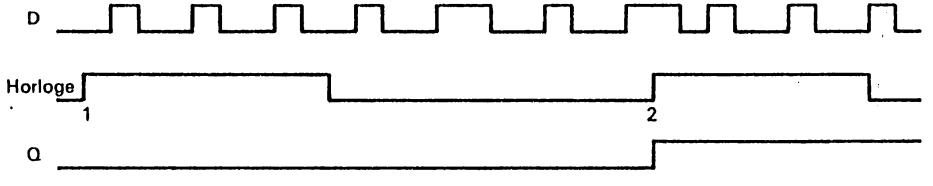


Fig. 3. 14

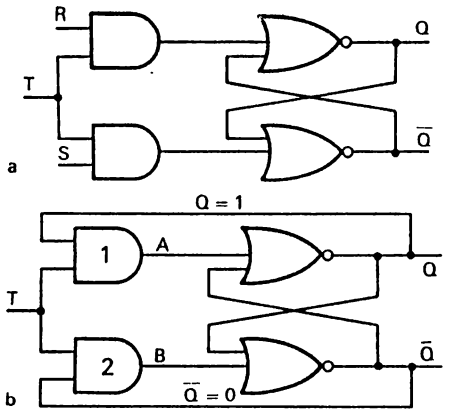


Fig. 3. 15

Une impulsion arrivant sur T, c'est la porte 1 qui va envoyer une impulsion en A, mettant Q à 0 et $\overline{Q}$ à 1. L'impulsion d'après doit inverser le phénomène.

En fait, si l'on essaie de faire fonctionner cette bascule, on n'obtient nullement l'effet désiré. Voyons pourquoi :

L'intervalle de temps entre les changements d'état des portes à l'entrée et à la sortie est de quelques dizaines de nanosecondes. Comme il y a de fortes chances que le créneau de T dure beaucoup plus longtemps, le basculement de Q et $\overline{Q}$ a le temps de se renouveler de nombreuses fois avant la fin du créneau, et l'état final est quelconque.

Pour remédier à cette situation deux solutions s'imposent :

— Réduire la commande sur T à une brève impulsion, de l'ordre de 20 ns. Un petit condensateur fera l'affaire. Associé à la résistance d'entrée des portes 1 et 2 il fera office de filtre passe-haut, transmettant une impulsion positive brève au moment du front montant du créneau sur T, puis une impulsion négative au moment du front descendant. Seule la première impulsion sera agissante.

— Retarder la réponse des sorties Q et $\overline{Q}$. Ici il faudra utiliser un filtre passe-bas RC.

On obtient alors la bascule T de la figure 3. 16 qui agit bien par basculements successifs à chaque créneau.

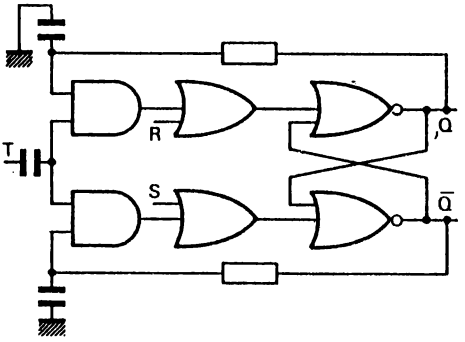


Fig. 3. 16

L'adjonction de deux portes OR permettra d'ajouter les commandes Set et Reset.

Une telle bascule n'est pas proposée en circuit intégré, car elle fait intervenir des condensateurs.

1.9.2 La bascule « maître-esclave »

En reprenant la même idée et en cherchant à supprimer les circuits RC, les électroniciens ont inventé la bascule « maître-esclave ».

Il s'agit en fait de deux bascules mises bout à bout. La première (bascule maître) réagira au front montant du créneau d'horloge, pendant ce temps la deuxième bascule sera inhibée. Puis, au front descendant du même créneau, la première bascule sera inhibée et la seconde agira en recopiant les sorties de la première.

Si l'on reboucle alors Q sur l'entrée R et $\overline{Q}$ sur l'entrée S, on obtiendra bien une bascule T, sans risque de basculements intempestifs... et sans circuits RC.

Cette bascule peut être réalisée avec des portes AND et NOR comme précédemment, ou bien avec des portes NAND comme sur la figure 3. 17,

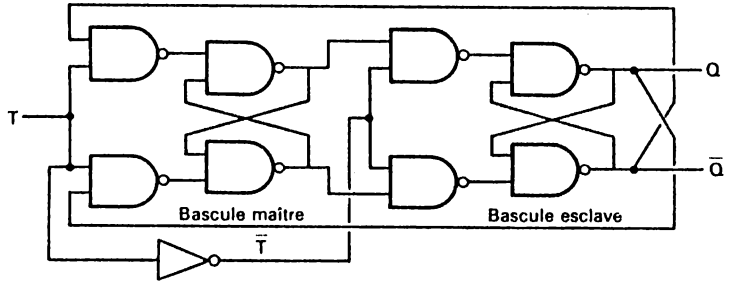


Fig. 3. 17

le principe restant toujours le même : la commande T agit par un front montant sur la bascule maître, tandis que son signal inversé inhibe la bascule esclave; au front descendant qui inhibe la bascule maître correspond un front montant sur la bascule esclave. Celle-ci recopie alors les sorties de la bascule maître.

Mais cette recopie s'effectue en inversant les valeurs, c'est pourquoi il faut croiser les fils de sortie avant de les ramener sur les portes NAND d'entrée.

Si vous essayez de construire une telle bascule avec des portes en circuits intégrés TTL, vous constaterez que les résultats sont décevants : le basculement ne s'opère pas franchement; il est le plus souvent précédé d'une série d'oscillations et la situation finale est quelquefois inchangée par rapport à la situation initiale.

En fait lorsque l'impulsion arrive sur la bascule maître, elle arrive aussi sur l'inverseur de la bascule esclave, mais le temps de basculement de cet inverseur laisse, pendant quelques nanosecondes, la bascule esclave en relation avec la bascule maître, et le même phénomène d'oscillations réapparaît.

Pour vaincre cet obstacle, les constructeurs ont fait appel à la technologie et ont réussi à mettre au point deux types de portes à seuil de déclenchement faible ou élevé.

Ainsi l'inverseur de T sur la bascule esclave est à seuil faible : il suffit que la tension atteigne de l'ordre de 1 V pour qu'il bascule. D'autre part les portes NAND d'entrée de la bascule maître sont à seuil élevé, de l'ordre de 3 V. Ainsi, pendant le temps de montée de l'impulsion de T, nous aurons deux instants t_1 et t_2 différents : à t_1 la bascule esclave sera isolée et à t_2 la bascule maître sera activée (*fig.* 3. 18).

Fig. 3. 18

Cette bascule T maître-esclave est un diviseur d'impulsions. Deux impulsions (au niveau 1) successives sur T se traduiront en sortie par une seule impulsion (*fig.* 3. 19). On appelle aussi cette bascule «diviseur par deux» ou «diviseur binaire».

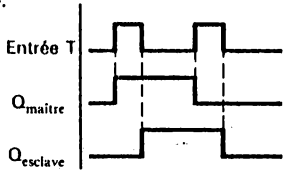


Fig. 3. 19

1.9.3 Autres types de bascules maître-esclave

Sur ce même principe les constructeurs proposent des bascules maître-esclave de type RS, de type D, ou de type JK.

Cette dernière bascule, très utilisée, s'obtient en ajoutant sur les NAND d'entrée de la figure 3. 17 un troisième fil, servant pour la commande J et la commande K.

J	K	Clk	Q	$\overline{Q}$
0	0	⎍	Q_0	$\overline{Q}_0$
1	0	⎍	1	0
0	1	⎍	0	1
1	1	⎍	$\overline{Q}_0$	Q_0

Fig. 3. 20

La table de vérité de ces bascules est donnée par la figure 3. 20.

On y voit que J = 1 et K = 0 permettent de mettre la sortie Q à 1, sur un créneau complet d'horloge.

Inversement, J = 0 et K = 1 mettront Q à 0 sur un créneau d'horloge.

La situation J = 1 et K = 1 ramène à la bascule T qui change de valeur après chaque créneau (diviseur par deux).

Ces actions sur J et K ne sont pas à confondre avec les commandes PRESET et CLEAR qui mettent Q à 1 ou 0 par impulsion sur ces fils, quelles que soient les valeurs de J, K ou T (Clk), et ce dès que l'impulsion parvient à la bascule.

Les bascules J K existent sous plusieurs versions en circuits intégrés. Assez souvent il y a 2 bascules par circuit. Selon les circuits, les deux bascules ont une horloge commune ou séparée, des entrées J et K ou $\overline{J}$, $\overline{K}$. Elles possèdent ou non des Preset et des Clear, communs ou séparés.

Certaines bascules peuvent être déclenchées par un front montant ou un front descendant à la place d'un créneau; ces bascules, commandées par un front, sont appelées dans la littérature anglo-saxonne «edge triggered».

Les bascules ont pour référence 70, 71, 72, 73, 74, 76, 101, 102, 103, 106, 107, 108, 109, 110, 111, 112, 113, 114.

Ce large éventail fait pressentir l'importance de ces bascules dans l'électronique digitale.

On retiendra aussi que la dénomination anglo-saxonne des bascules est «flip-flop», souvent écrit FF en abrégé.

2. MONOSTABLES

Une bascule monostable (on dit aussi multivibrateur monostable) est une bascule dont la sortie Q, après avoir basculé sous l'effet d'une impulsion de commande, revient spontanément à sa valeur d'origine au bout d'un certain délai. Ce délai est obtenu par l'insertion d'un couple RC.

2.1 Montage avec inverseurs

Le montage correspond à la figure 3. 21. Sous l'effet d'un créneau positif sur l'inverseur 1, le condensateur C se décharge à travers la diode D (cette diode coupe la communication dès que l'impulsion de commande cesse). L'inverseur 2 bascule alors et Q = 1.

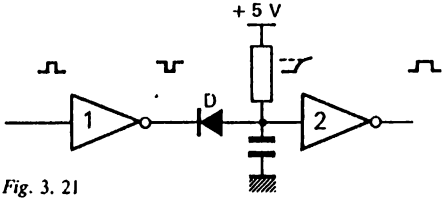


Fig. 3. 21

Par la résistance R le condensateur se recharge et, dès que le seuil de tension est atteint, Q passe à 0.

La technologie TTL ne permet pas d'atteindre des délais très importants. R doit être limitée à environ 10 kΩ, et le temps de recharge du condensateur est beaucoup plus rapide que la valeur RC ne le laisserait penser.

Les résultats sont plus probants avec les circuits CMOS et il est recommandé d'utiliser des inverseurs d'un boîtier 74C04, par exemple. On n'oubliera pas que la sortie ne peut être reliée à des portes TTL, sinon les modèles LS ou L.

L'inverseur 2 n'est pas soumis à une impulsion franche, puisque la tension monte suivant la courbe exponentielle de charge d'un condensateur. Entre le seuil de déclenchement bas et le seuil haut, une zone d'« incertitude » risque de donner, en sortie, un niveau tout aussi incertain, et quelquefois des basculements intempestifs.

Pour éviter cet inconvénient, on aura intérêt à utiliser un « trigger de Schmitt ».

2.2 Trigger de Schmitt

Lorsqu'une impulsion de commande risque d'arriver affaiblie ou mal formée sur une commande, il est bon non seulement de la reconformer, mais encore d'éviter des ressauts intempestifs.

Observez la figure 3. 22 *a*. Si le seuil de basculement de la porte sur laquelle arrive cette impulsion est e, la porte risque de s'ouvrir à t_1, de se refermer à t_2 et de s'ouvrir à nouveau à t_3, alors qu'il ne devrait y avoir qu'une seule ouverture.

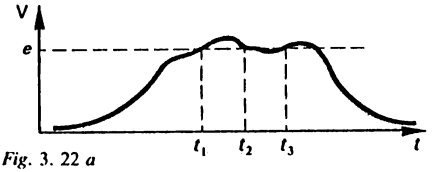


Fig. 3. 22 *a*

Pour éviter cela, il serait utile d'avoir un seuil (de tension) d'ouverture e_1 plus élevé que le seuil de tension de fermeture e_2 (*fig.* 3. 22 *b*). La même impulsion serait alors prise en compte une seule fois malgré son sommet irrégulier.

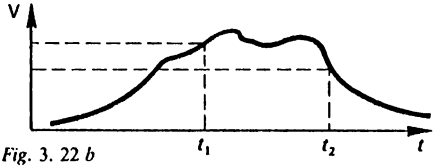


Fig. 3. 22 *b*

Le trigger de Schmitt répond à ce problème.

On peut le réaliser en composants discrets suivant le schéma de la figure 3. 23.

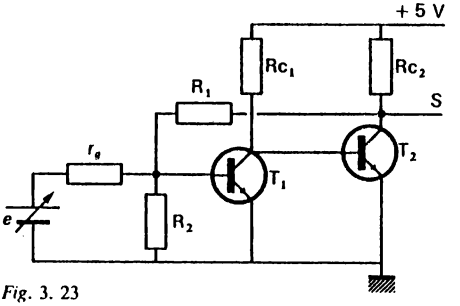


Fig. 3. 23

Pour un signal 0 ($e = 0$), T_1 est bloqué et T_2 saturé, S = 0. Tant que le seuil du transistor T_1 n'est pas franchi, S reste à 0.

R_1 et R_2 étant en parallèle (S à la masse), la tension V_A à l'entrée de T_1 est donnée par la formule (*fig.* 3. 24)

$$V_A = \frac{e(R_1 // R_2)}{r_g + (R_1 // R_2)} = \frac{e R_1 R_2}{r_g(R_1 + R_2) + R_1 R_2}$$

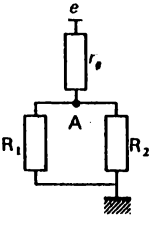


Fig. 3. 24

L'amplificateur basculera pour $V_A = 0{,}7$ volt. Pour que ceci corresponde à $e = 1$ volt (par exemple), il faut donc que

$$1 = \frac{0{,}7(r_g R_1 + r_g R_2 + R_1 R_2)}{R_1 R_2} \qquad (1)$$

Dès que ce seuil est franchi, S passe à 5 V.

Au point A, l'action du générateur e se superpose alors à celle de la sortie S. La tension au point A s'obtient par le théorème de superposition (*fig.* 3.25)

$$V_A = V_{A(e)} + V_{A(E)} = \frac{e R_1 R_2}{r_g R_1 + r_g R_2 + R_1 R_2} + \frac{E r_g R_2}{R_1 r_g + R_1 R_2 + R_2 r_g}$$

$$V_A = \frac{e R_1 R_2 + E r_g R_2}{r_g R_1 + r_g R_2 + R_1 R_2}$$

$$\text{et } e = \frac{V_A(r_g R_1 + r_g R_2 + R_1 R_2) - E R_2 r_g}{R_1 R_2}$$

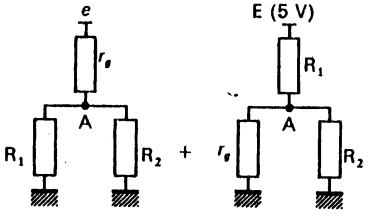


Fig. 3. 25

Si l'on souhaite que le nouveau seuil de basculement ($V_A = 0{,}7$ V) soit atteint pour $e = 0{,}5$ V, il faudra :

$$0{,}5 = \frac{0{,}7(r_g R_1 + r_g R_2 + R_1 R_2)}{R_1 R_2} - \frac{E R_2 r_g}{R_1 R_2} \qquad (2)$$

Notons tout de suite que

$$e_2 = e_1 - \frac{E R_2 r_g}{R_1 R_2} = e_1 - \frac{E r_g}{R_1} \qquad (3)$$

c'est-à-dire que le seuil de basculement à la retombée du signal est bien inférieur au seuil de basculement à la montée du signal. On a ici un effet d'hystérésis bien visible sur le schéma de la figure 3. 26.

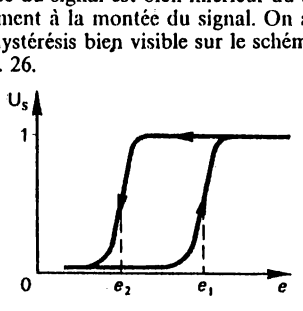


Fig. 3. 26

Avec $r_g = 10$ kΩ, $R_{c_1} = R_{c_2} = 1$ kΩ nous obtiendrons

– avec la relation (3) : $\frac{E r_g}{R_1} \triangleq e_1 - e_2$

$$\frac{5 \times 10^4}{R_1} = 0{,}5 \rightarrow \boxed{R_1 = 100 \text{ k}\Omega}$$

– avec l'expression (1) on obtient alors

$$\boxed{R_2 = 30 \text{ k}\Omega}$$

La bascule de Schmitt est proposée, par les fabricants de circuits intégrés, dans des versions d'inverseurs (7414) ou de portes NAND à quatre entrées (7413) ou à deux entrées (74132).

Ces portes se reconnaissent au schéma simplifié des catalogues qui porte le symbole du cycle d'hystérésis au centre du symbole de la porte (*fig.* 3. 27).

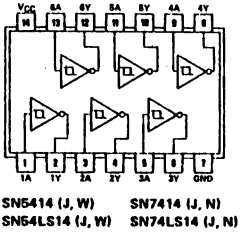


Fig. 3. 27 (Doc. Texas Instruments)

Ce symbole est inversé par rapport à celui de la figure 3. 26 lorsqu'il s'agit de portes inverseuses (inverseurs, portes NAND, etc.).

Les catalogues des fabricants indiquent le seuil V_{T+} lorsque la tension d'entrée s'élève, et le seuil V_{T-} lorsque celle-ci retombe.

Pour une TTL, les valeurs nominales sont de 1,6 V et 0,8 V, soit une différence de 0,8 V pour une alimentation de 5 V et une tension de sortie nominale de 3,4 V.

On appréciera, dans ce domaine, les performances de l'inverseur avec trigger de Schmitt de la série CMOS : le 74C14 de même brochage que le précédent. Pour ce composant, les valeurs V_{T+} et V_{T-} sont respectivement de 3,6 V et de 1,4 V, pour une sortie de 5 V, c'est-à-dire que ces seuils sont parfaitement symétriques par rapport à la tension moyenne de 2,5 V, et leur écart est bien plus net (2,2 V au lieu de 0,8 V).

2.3 Monostables en circuits intégrés

Le plus utilisé des monostables, pour des temporisations courtes (fraction de seconde), est le circuit 121. Il est nécessaire de lui adjoindre le

circuit RC en éléments discrets, mais il possède son propre trigger de Schmitt et peut être commandé par deux entrées sur front descendant (A_1 et A_2) ou par une entrée sur front montant (*fig.* 3. 28).

FUNCTION TABLE

INPUTS			OUTPUTS	
A1	A2	B	Q	$\bar{Q}$
L	X	H	L	H
X	L	H	L	H
X	X	L	L	H
H	H	X	L	H
H	↓	H	⎍	⎎
↓	H	H	⎍	⎎
↓	↓	H	⎍	⎎
L	X	↑	⎍	⎎
X	L	↑	⎍	⎎

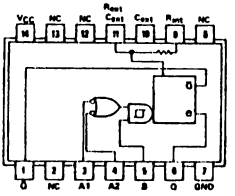


Fig. 3. 28 (Doc. Texas Instruments)

Par un système interne de compensation, la durée de la temporisation est rendue insensible aux différences de tension d'alimentation et de température.

On trouve d'autres monostables avec quelques variantes dans les circuits 74122, 123 et un double monostable dans le 221.

2.4 Utilisation en monostable du circuit 555

2.4.1 Morphologie du 555

Le circuit 555 est un outil polyvalent de l'électronique. Il comporte une bascule SR dont les entrées sont commandées par deux amplis-comparateurs formant ce que l'on appelle un comparateur à fenêtre.

On voit sur la figure 3. 29 ces comparateurs branchés sur un pont de résistances. Les trois résistances de 5 kΩ, dotées d'une grande précision, divisent la tension d'alimentation par trois.

L'inverseur inférieur reçoit, par son entrée « plus » une tension de $V_{CC}/3$. Un fil de commande « trigger » est branché sur l'entrée « moins ».

L'inverseur supérieur est branché au point de tension $2V_{CC}/3$ par son entrée « moins » et une commande « threshold » (seuil) aboutit à son entrée « plus ».

On aura donc une action sur R si la tension du « trigger » passe au-dessous de $V_{CC}/3$ et une action sur S si la tension du « threshold » passe au-dessus de $2V_{CC}/3$.

La sortie Q de la bascule SR sert d'entrée d'une part à un inverseur de puissance moyenne dont la sortie est nommée « Out » et d'autre part à un transistor, également de moyenne puissance, dont le collecteur ouvert est nommé « Discharge » (le courant pouvant traverser la sortie est de l'ordre de 200 mA).

2.4.2 Montage du 555 en monostable

Ce montage se fait suivant le schéma de la figure 3. 30.

Un couple RC est monté entre alimentation et masse, son point milieu est relié à l'entrée threshold. Cette même entrée reçoit le collecteur « discharge » du transistor de sortie.

Une impulsion négative sur le trigger met la sortie Q à 0 (« Out » à 1) et bloque le transistor. Le condensateur se charge suivant une courbe exponentielle à travers la résistance R. Lorsque la tension sur threshold atteint 2/3 de V_{CC}, la sortie Q passe à 1, le transistor s'ouvre et ramène rapidement la tension sur threshold à 0.

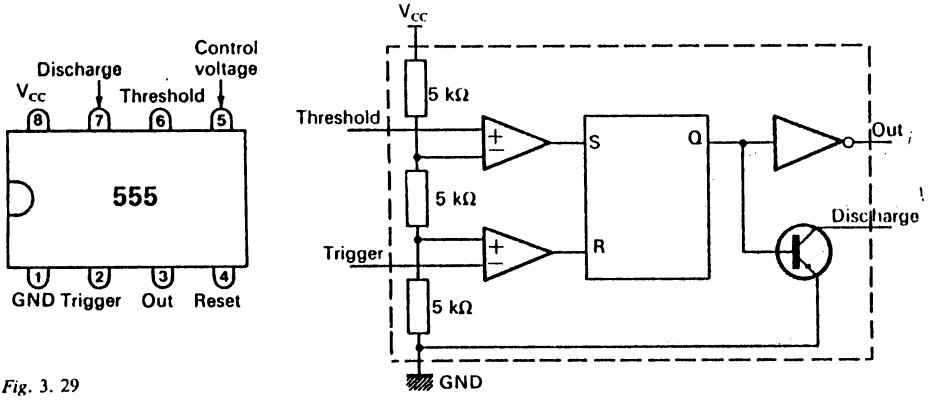


Fig. 3. 29

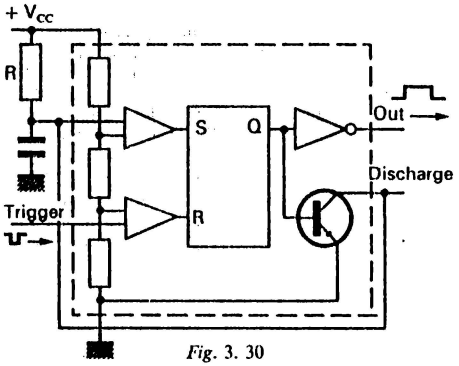


Fig. 3. 30

Tant qu'aucun nouveau créneau négatif ne parvient sur le trigger, la situation reste la même (Q = 1, out = 0, discharge à 0).

Ainsi équipé, le 555 joue donc bien le rôle d'un monostable, commandé par un front descendant sur l'entrée trigger. En sortie Out on aura un front montant quasi-instantané, puis, au bout d'un délai égal à 1,1 RC, un passage au niveau bas.

Les temporisations que l'on peut obtenir avec un tel montage peuvent atteindre facilement la minute sans risque de détérioration du circuit.

3. LES MULTIVIBRATEURS ASTABLES

Ces bascules méritent bien le nom de multivibrateurs car elles émettent une succession de créneaux alternativement hauts et bas. L'utilisation de réseaux RC permet très généralement de les obtenir.

3.1 Le plus simple des multivibrateurs astables

Il est obtenu en faisant appel au circuit C.MOS 74C14 présenté ci-dessus (6 inverseurs à trigger de Schmitt).

Il suffit de brancher une résistance R entre la sortie et l'entrée de cet inverseur et un condensateur entre entrée et masse (*fig.* 3. 31).

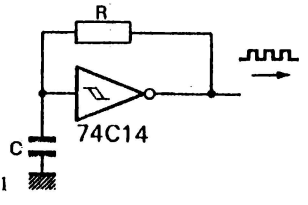


Fig. 3. 31

— Si l'entrée est à 0, la sortie au niveau haut va charger le condensateur à travers R. La tension monte progressivement sur l'entrée jusqu'à ce qu'elle atteigne V_{T+}.

A ce moment-là, l'inverseur bascule et C commence à se décharger à travers R jusqu'à ce que l'entrée atteigne le seuil V_{T-}.

La symétrie, en C.MOS, des deux seuils, permet d'obtenir en sortie des signaux carrés de forme régulière. La période de ce signal est d'environ 1,7 RC.

La même réalisation est possible avec un circuit TTL 7414, mais cette fois, étant donnée l'asymétrie des seuils V_{T+} et V_{T-}, le signal carré aura un créneau positif très bref et un créneau négatif plus long (2 à 3 fois). On dit que son «rapport cyclique» est de 30 % si le créneau positif a une durée de 30 % de la période.

3.2 Circuit RC et inverseur normal

Si l'on dispose, à l'entrée d'un inverseur, d'un circuit RC branché comme l'indique la figure 3. 32 *a* on observe les phénomènes suivants :

A l'état de repos, quel que soit le potentiel de A (0 ou 1), B est au niveau bas, donc S au niveau haut.

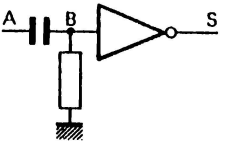


Fig. 3. 32 *a*

Supposons alors que A soit à 1. Si cette entrée passe brusquement à 0, une impulsion négative (< 0) apparaît sur B. Cette impulsion est très courte, car l'entrée de l'inverseur est équivalente à une diode orientée de la masse vers cette entrée. De toute façon S reste au niveau haut (chronogramme de la *fig.* 3. 32 *b*).

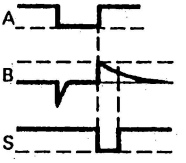


Fig. 3. 32 *b*

Si A passe à nouveau au niveau haut, «B» passe à 1 puis sa tension décroît sous l'effet de la décharge du condensateur dans R (la résistance d'entrée de l'inverseur est grande). S passe alors à 0 pour reprendre la valeur 1 lorsque le seuil de basculement de l'inverseur est atteint.

En bref, le front haut sur A a généré une impulsion négative sur S.

3.3 Double circuit RC + inverseur

En ajoutant un deuxième élément identique au précédent on obtient le circuit de la figure 3. 33 *a*.

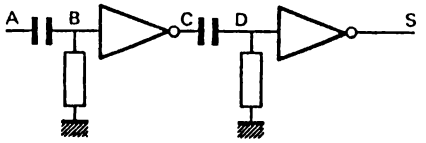


Fig. 3. 33 *a*

Il est aisé de construire le chronogramme à cinq traces des variations de tension en A, B, C, D et S (*fig.* 3. 33 *b*).

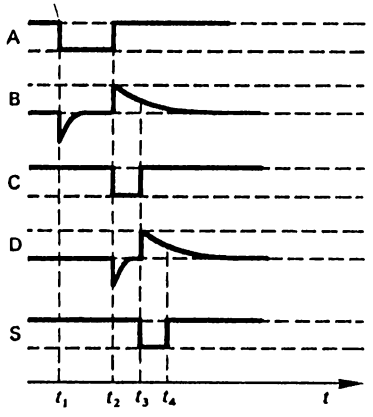


Fig. 3. 33 *b*

Le passage du front haut sur A à l'instant t_2 a donc généré une impulsion de durée $t_3 - t_2$ sur C, puis une impulsion de durée $t_4 - t_3$ sur S.

Si les deux circuits RC sont identiques, ces deux impulsions ont même durée.

3.4 Schéma du multivibrateur

Si l'on branche la sortie S du montage précédent sur l'entrée A (*fig.* 3. 34), le décalage apparu entre le créneau négatif de A et le créneau négatif de S sur la figure 3. 33 *b*) va, avec un tel rebouclage, créer des oscillations. On a obtenu un multivibrateur astable.

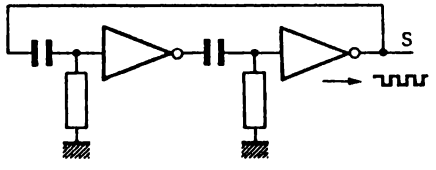


Fig. 3. 34

Si les deux circuits RC sont identiques, les durées des créneaux négatifs et positifs seront identiques.

En fait la forme de ces créneaux n'est pas très « carrée » car ils subissent l'influence des charges et décharges successives des circuits RC. Il suffira de disposer d'un inverseur supplémentaire en sortie pour les reconformer.

3.5 Réalisation pratique

Ce multivibrateur est simple à réaliser avec un circuit intégré du type 04 et on comprendra qu'il n'ait pas donné lieu à un circuit intégré spécial, puisqu'on doit de toute façon y implanter les condensateurs et résistances correspondant aux fréquences à obtenir.

Il est bon, si l'on veut obtenir de bons résultats, d'utiliser le circuit 74LS04 à base de transistors et diodes Schottky, circuit qui met en jeu de faibles courants et présente de faibles capacités parasites d'entrée. Il faudra, de toute façon, choisir des condensateurs présentant un faible courant de fuite.

Ces précautions étant prises, les fréquences obtenues pourront aller de quelques hertz à quelques mégahertz, la mise sous tension suffisant à provoquer le déclenchement des impulsions.

Une autre possibilité est d'utiliser des composants discrets : il suffit de se rappeler qu'un inverseur équivaut à un transistor monté en émetteur commun.

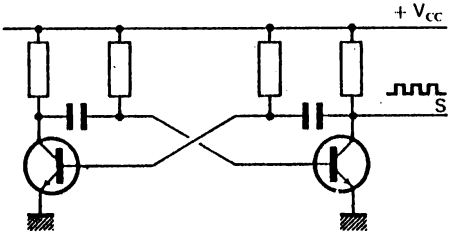


Fig. 3. 35

En plaçant alors les résistances du couple RC non plus à la masse mais au pôle + pour qu'elles servent en même temps à la polarisation des bases des transistors, on obtient le schéma de la figure 3. 35, schéma classique de ce que l'on appelle le multivibrateur d'Abraham-Bloch, dont l'invention remonte à l'époque des tubes à vide.

3.6 Utilisation du 555 en multivibrateur astable

Il n'est pas inintéressant d'employer un circuit 555 pour jouer le rôle de multivibrateur astable : son courant de sortie important pourra alimenter sans problème un grand nombre de portes.

Il existe plusieurs possibilités de montage. Une des plus simples est présentée figure 3. 36. Ce montage permet d'obtenir un bon « facteur de forme », c'est-à-dire des signaux dont les créneaux positifs et négatifs sont à peu près de même durée.

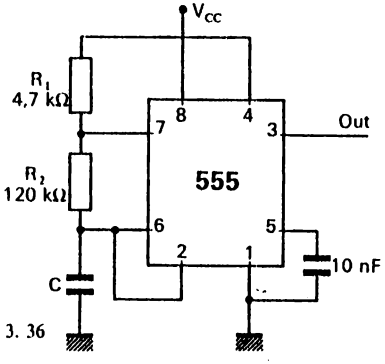


Fig. 3. 36

Pour cela, la résistance R_1 sera de 4,7 kΩ et la résistance R_2 de 120 kΩ. Le choix du condensateur C permettra de fixer la fréquence des créneaux de sortie (de quelques dizaines d'hertz à près de 500 000 Hz).

3.7 Autres réalisations

On pourrait étendre à l'infini le nombre de possibilités d'obtention d'un multivibrateur astable. Tous les oscillateurs de l'électronique analogique peuvent être utilisés. Il suffit de saturer le signal de sortie pour obtenir des signaux carrés. On peut aussi utiliser un trigger de Schmitt, ce qui constitue une solution plus élégante.

Parmi tous ces oscillateurs, l'électronique digitale retient essentiellement ceux qui permettent des fréquences élevées et stables, car c'est à leur rythme que se feront les traitements d'information, et il sera quelquefois important, notamment dans les télécommunications, qu'il y ait un bon synchronisme entre oscillateurs différents.

Pour ces deux raisons, ce sont souvent des quartz, enfermés dans des enceintes isothermes, qui sont utilisés.

Nous ne passerons pas en revue les différentes horloges à quartz, les constructeurs proposant des circuits intégrés complets ou adaptables par simple adjonction d'un quartz extérieur. Les fréquences obtenues sont souvent dans la zone de 1 à 20 MHz.

EXERCICES

Les exercices qui sont proposés ici revêtent l'aspect d'expériences à faire avec les circuits intégrés classiques du commerce et notamment les bascules D du circuit 7474.

1. Réalisation d'un chenillard.

Le chenillard est une guirlande d'ampoules où l'on a l'impression de voir la lumière se propager d'une lampe à l'autre grâce à des séries bien calculées d'allumages et d'extinctions. Un chenillard peut comporter plusieurs dizaines de lampes, mais son principe de base repose sur une séquence répétée plusieurs fois, intéressant quelques lampes seulement (trois au minimum). Nous allons réaliser cette séquence de base sur quatre lampes ou plutôt quatre L.E.D., son extension étant toujours possible à partir des mêmes principes de réalisation.

1° *Description du chenillard à réaliser* (*fig.* E. 3. 1) : A l'instant t_0 nous voulons que les L.E.D. 1, 5 et 9 soient allumées et toutes les autres éteintes.

A l'instant t_1, ce sont les L.E.D. 2, 6 et 10 qui devront être allumées, puis à l'instant t_2 les L.E.D. 3, 7 et 11, enfin à l'instant t_3 les L.E.D. 4, 8 et 12.

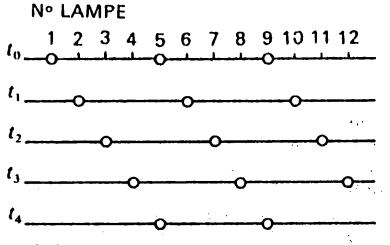


Fig. E. 3. 1

A l'instant t_4 nous obtiendrons la même configuration qu'à l'instant t_0.

On obtient donc le chronogramme des signaux qui doivent parvenir sur les L.E.D. 1, 5, 9 d'une part, les L.E.D. 2, 6, 10, etc. (*fig.* E. 3. 2).

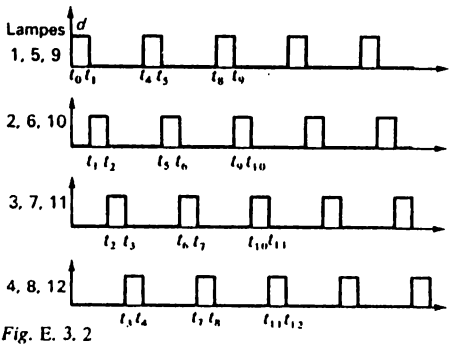


Fig. E. 3. 2

Chaque signal est décalé par rapport au précédent de $\frac{t_4 - t_0}{4}$, et ce décalage correspond aussi à la durée des créneaux positifs de chaque signal.

2° *Réalisation :* On pourrait imaginer quatre générateurs de signaux parfaitement identiques et se déclenchant en cascade, ce qui laisse supposer de nombreux réglages du fait de l'imperfection des composants dont on dispose. Il est plus sage de ne faire appel qu'à un seul générateur (une même horloge).

Imaginons donc un signal d'horloge parfaitement carré (dont les créneaux positifs et les créneaux négatifs ont la même durée). Faisons alors parvenir ce signal sur une bascule D dont l'entrée D est à 1 (*fig.* E. 3. 3).

Au premier front positif de l'horloge, la sortie Q de cette bascule passera à 1 (en supposant qu'elle a été remise à 0 au préalable) (voir *fig.* E. 3. 4).

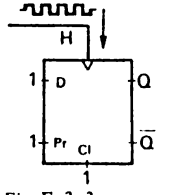


Fig. E. 3. 3

Fig. E. 3. 4

Si l'on met alors en série plusieurs de ces bascules, l'entrée D de chacune étant reliée à la sortie Q de la précédente et l'horloge étant commune à chacune d'elles, on obtiendra le montage de la figure E. 3. 5 et le chronogramme E. 3. 6.

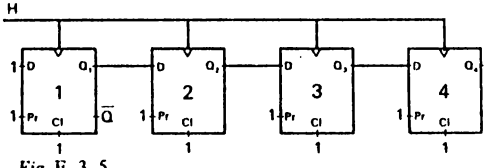


Fig. E. 3. 5

On peut se demander pourquoi, par exemple, la bascule 2 ne bascule pas en même temps que la 1 puisque le front d'horloge l'atteint au moment où Q_1 passe à 1 (donc D est aussi à 1). Il n'en est rien pour une raison qui tient à quelques nanosecondes : lorsque le front haut arrive sur 1 il s'écoule en effet ces quelques nanosecondes (délai d*t*) avant que Q ne passe à 1 (*fig.* E. 3. 7). Cela suffit pour que 2 ne bascule pas.

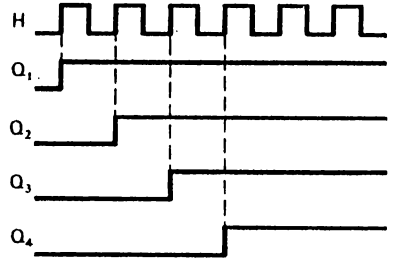


Fig. E. 3. 6

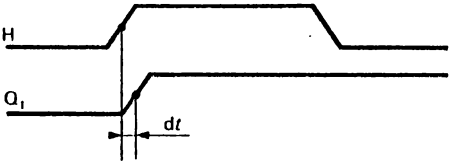


Fig. E. 3. 7

Il faut maintenant obtenir le retour à 0 de chaque sortie Q_1 à Q_4. Il suffit pour cela de reboucler Q_4 sur D_1 et d'amorcer au préalable la séquence par un ordre «Clear» sur les bascules 2 à 4 et un «Preset» sur la bascule 1.

3° *Construction pratique.*

L'utilisation de L.E.D. permet de voir fonctionner le montage sans le secours d'un oscilloscope. Pour réaliser un vrai chenillard, il faudra avoir recours aux moyens d'amplification classiques, mais là n'est pas notre sujet.

a) L'horloge : Pour bien visualiser le phénomène, l'horloge doit être lente. Il est possible de s'inspirer des oscillateurs proposés dans ce chapitre (double inverseur + RC, 555 etc.). Le circuit 7414 permet une réalisation simple avec une résistance de 1,2 kΩ (ne pas aller au-delà) et un condensateur de 150 µF (voir *fig.* E. 3. 8).

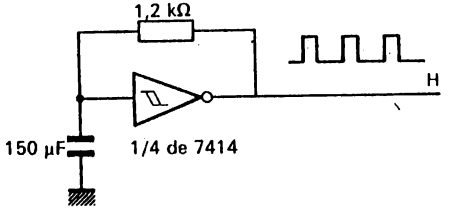


Fig. E. 3. 8

Seul problème : le «facteur de forme» des oscillations est de l'ordre de 30 % (le créneau positif dure 30 % de la période et le créneau positif 70 %).

Ceci n'a pas d'inconvénient dans notre réalisation, car nous n'allons utiliser que les fronts montants de ces créneaux d'horloge. Or ces fronts montants sont régulièrement espacés les uns des autres.

b) Les bascules : On peut utiliser les circuits 7474 contenant chacun deux bascules D. (Il faudra donc, en tout, deux circuits de ce type.)

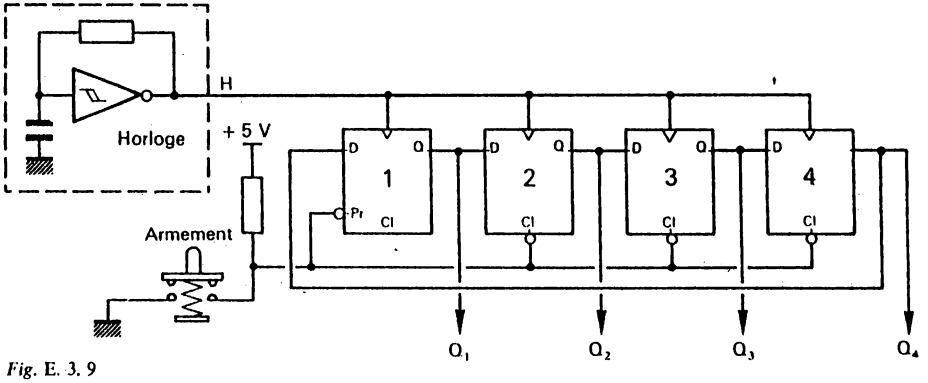


Fig. E. 3. 9

Le câblage doit être fait conformément à la figure E. 3. 9 sans oublier le circuit d'armement. On notera à ce propos que les commandes Preset et Clear de ces circuits sont précédées d'un petit rond, leur action s'exerce donc par un créneau négatif.

c) L'affichage à diodes : Le plus simple est d'utiliser ici un circuit intégré à collecteur ouvert. Les diodes, accompagnées d'une résistance de 330 Ω, seront mises en place sur ce collecteur. Comme vous le savez, l'allumage des diodes correspond en fait à un signal 0 sur la sortie de la porte, et leur extinction à un signal 1. Il faut donc utiliser des circuits inverseurs.

Le circuit 7401 offre la possibilité de brancher les quatre L.E.D. (*fig.* E. 3. 10).

A la mise sous tension on obtient n'importe quoi. Il suffit d'appuyer sur le bouton d'armement pour que le chenillard se mette en route.

On pourra compléter ce chenillard par deux autres séries de L.E.D. en disposant deux autres circuits 7401 branchés sur les mêmes sorties Q_1 à Q_4.

2. Diviseur par un nombre pair quelconque.

En utilisant quelques principes du chenillard précédent, on peut imaginer un circuit qui diviserait la fréquence d'une horloge H par un nombre pair quelconque.

Imaginez par exemple le montage qui permet de transformer le signal H de l'horloge en un signal de fréquence dix fois plus faible, à facteur de forme de 50 %.

Il faut donc que chaque créneau positif ou négatif du signal de sortie ait une durée de cinq périodes du signal H d'origine.

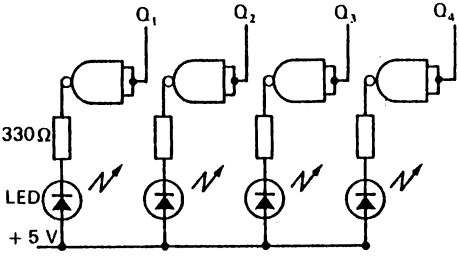


Fig. E. 3. 10

CHAPITRE 4

Les compteurs et la numération binaire

1. COMPTER AVEC DES BASCULES... MAIS JUSQU'OÙ?

Nous avons vu, au § 6 du chapitre 2, le système de numération appelé binaire pur. Il est basé sur le même principe que le système décimal, mais alors que ce dernier comporte dix signes (de 0 à 9) le système binaire pur n'en comporte que deux : 0 et 1. Lorsque le rang des unités a utilisé tous les signes, on crée un deuxième rang, puis un troisième, etc... pour y porter les puissances de deux successives (et dans le système décimal, les puissances de dix).

Rappelons-nous la composition des 18 premiers nombres par exemple (*fig.* 4. 1).

Pour l'électronicien, il est fort instructif de lire non pas les lignes correspondant à chaque nombre binaire, mais chacune des colonnes correspondant aux unités de rang 1, 2, 4, 8 et 16.

En lisant ces colonnes à partir du haut, on constate les points suivants :

Dans la première colonne (à droite), les 0 et les 1 alternent régulièrement. Une bascule astable

Système décimal			Système binaire pur				
10^1	10^0 (unités)		2^4	2^3	2^2	2^1	2^0 (unités)
0	0	→	0	0	0	0	0
0	1		0	0	0	0	1
0	2		0	0	0	1	0
0	3		0	0	0	1	1
0	4		0	0	1	0	0
0	5		0	0	1	0	1
0	6		0	0	1	1	0
0	7		0	0	1	1	1
0	8		0	1	0	0	0
0	9		0	1	0	0	1
1	0		0	1	0	1	0
1	1		0	1	0	1	1
1	2		0	1	1	0	0
1	3		0	1	1	0	1
1	4		0	1	1	1	0
1	5		0	1	1	1	1
1	6		1	0	0	0	0
1	7		1	0	0	0	1
1	8		1	0	0	1	0
		... etc...					

Fig. 4. 1

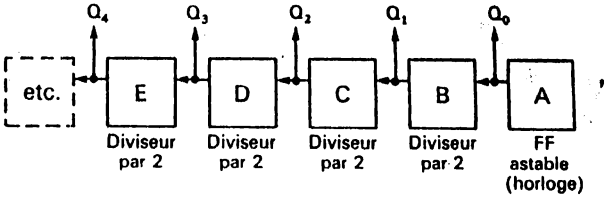


Fig. 4. 2

peut être utilisée pour fournir cette série de valeurs.

Dans la deuxième colonne il y a alternance de deux 0 et de deux 1. On pourra réaliser cette série par une bascule câblée en « diviseur par deux » branchée sur la bascule précédente.

Dans la troisième colonne le nombre de 0 et de 1 qui alternent double par rapport à celui de la colonne précédente.

C'est le même phénomène qui peut être observé dans chaque colonne suivante.

On peut obtenir ces valeurs par de nouvelles bascules montées en diviseurs par deux raccordées aux bascules précédentes. Avec un grand nombre de bascules on pourra compter jusqu'à un « très grand nombre » (*fig.* 4..2).

Mais si l'on veut disposer d'organes compteurs, il est important de savoir à l'avance jusqu'où ils auront à compter.

Par ailleurs la mise en série de nombreuses bascules pose quelques problèmes.

Ces aspects vont être étudiés dans les paragraphes qui suivent.

2. LE CODE BCD

2.1 Principe du BCD

La machine étant faite pour l'homme et non l'homme pour la machine, la solution la plus évidente, sinon celle qui a été trouvée en premier lieu, a été de rattacher étroitement le système binaire au système décimal en créant des « unités de comptage » capables de compter de 0 à 9 et non au-delà.

Le 9 (voir *fig.* 4. 1) correspondant au nombre binaire 1001, il faudra 4 cellules binaires pour jouer ce rôle. On dit aussi qu'il faut 4 bits pour représenter un chiffre décimal (bit est le raccourci de « binary digit »).

Si nous voulons coder le nombre 1729, l'utilisation du système à 4 bits nous donnera 0001 - 0111 - 0010 - 1001. (Notez bien qu'il faut toujours 4 bits, même si ce sont des zéros.)

Cette méthode de représentation des nombres décimaux par leur codage respectif en binaire s'appelle le système BCD, de l'anglais « Binary Coded Decimal » qui veut dire Décimal Codé en Binaire!

Ce système facilite pour le moins la lecture, si elle est nécessaire, de nombres binaires importants.

En binaire pur, le même nombre 1729 que nous avons choisi précédemment, se traduirait par

$$11011000001 \qquad (1)$$

En BCD, il est représenté par

$$0001 - 0111 - 0010 - 1001 \qquad (2)$$

La formule (1) est intraduisible si l'on ne dispose pas de crayon et de papier pour faire sa transcription, tandis que la formule (2) se traduit assez facilement dès que l'on a mémorisé les 10 premiers nombres binaires.

On peut, de plus, imaginer aisément que la traduction automatique, par des circuits électroniques de décodage, sera plus facile à réaliser dans le cas (2) que dans le cas (1).

En revanche nous constatons qu'il faudra, en BCD, utiliser plus d'éléments binaires, donc plus de circuits, que dans le cas (1).

Étudions aussi les problèmes que vont poser les opérations simples appliquées à ce code BCD.

2.2 L'addition en BCD

La table de toutes les additions binaires
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 (0, et 1 de retenue)

Fig. 4. 3

Choisissons un cas simple : l'addition de 8 et de 4 (les nombres décimaux seront inscrits avec un 10 en indice)

8_{10} correspond en BCD à 1000

4_{10} correspond en BCD à 0100

Si nous appliquons les lois de l'addition binaire représentées sur la figure 4. 3, nous obtiendrons

$$8_{10} + 4_{10} = 12_{10} = 1100$$

ceci est bien vrai en binaire pur, mais n'est plus vrai en BCD où 12_{10} doit s'écrire 0001 - 0010.

Ceci est dû au fait que l'addition de 8 et 4 génère une retenue de rang supérieur (8 + 4 = 2 plus 1 de retenue).

Cette valeur 10_{10}, qui se traduit par 1010 en binaire pur, est donc à retrancher de la valeur 1100 obtenue, pour venir se loger sous forme d'une unité dans la première cellule de 4 bits d'ordre supérieur :

$$1100 - 1010 = 0010$$

Il faudrait donc que la machine qui calcule en BCD fasse, à chaque opération élémentaire, un contrôle du résultat obtenu (le nombre est-il ou non supérieur à 9, si oui ôter 10, reporter 1 sur le groupe de rang supérieur, etc.).

Cet ensemble de contrôles et d'opérations est long et coûteux en circuits supplémentaires. C'est pourquoi on l'utilise peu dans des organes de calculs.

Le BCD reste essentiellement le code du dialogue entre l'homme et la machine.

2.3 L'affichage sept segments

Une fois les nombres binaires traduits en BCD, il est possible d'aller plus loin et de transformer chaque bloc de 4 bits en un chiffre directement lisible par l'homme.

L'affichage de ce chiffre se fera grâce à un « afficheur sept segments » bien connu aujourd'hui puisqu'on le trouve aussi bien sur le cadran des montres que sur le tableau de bord des automobiles (*fig.* 4. 4).

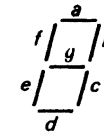
Fig. 4. 4

Il comprend en effet sept segments qui sont souvent des diodes électroluminescentes de forme allongée; ces segments, disposés en forme de 8 permettent de représenter à peu près correctement tous les chiffres de 0 à 9.

2.3.1 *Réalisation du décodeur BCD 7 segments*

Comment traduire la série des valeurs représentées par les quatre bits du code BCD en sept valeurs correspondant aux sept segments de l'afficheur?

En donnant un nom à chacun des sept segments (de « a » jusqu'à « g »), la correspondance entre le chiffre à afficher et les segments à allumer s'obtient aisément. Elle est donnée par la figure 4. 5.



En appelant d'autre part DCBA les quatre bits du nombre BCD, en partant du bit de poids le plus fort, nous pouvons bâtir un tableau de Karnaugh indiquant, dans chaque case, quel chiffre devra être affiché en fonction des valeurs de ces quatre bits (*fig.* 4. 6).

Chiffre	Affichage	*a*	*b*	*c*	*d*	*e*	*f*	*g*
0		1	1	1	1	1	1	0
1		0	1	1	0	0	0	0
2		1	1	0	1	1	0	1
3		1	1	1	1	0	0	1
4		0	1	1	0	0	1	1
5		1	0	1	1	0	1	1
6		1	0	1	1	1	1	1
7		1	1	1	0	0	0	0
8		1	1	1	1	1	1	1
9		1	1	1	1	0	1	1

Fig. 4. 5

Chiffre décimal	Correspondance BCD			
	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

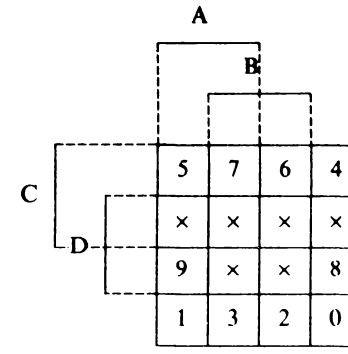


Fig. 4. 6

Les cases marquées d'une croix correspondent à des cas exclus. Nous pourrons les utiliser à notre guise.

En mettant en relation les figures 4. 5 et 4. 6 nous voyons, par exemple, que le segment *a* ne doit rester éteint que pour les chiffres 1 et 4. Ceci correspond au tableau de Karnaugh de la figure 4. 7.

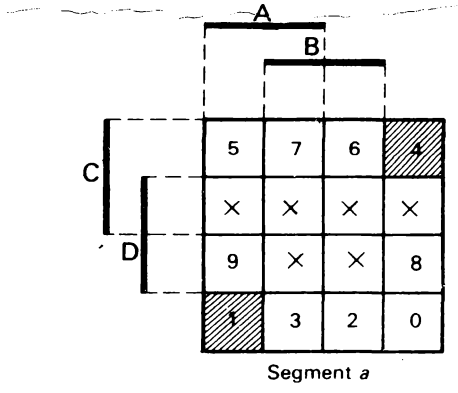


Fig. 4. 7

Comme nous avons le droit de disposer à notre guise des cases marquées d'une croix, les domaines où *a* doit être allumé peuvent se délimiter comme sur la figure 4. 8, et nous obtenons l'expression

$$a = AC + \bar{A}\bar{C} + B + D$$

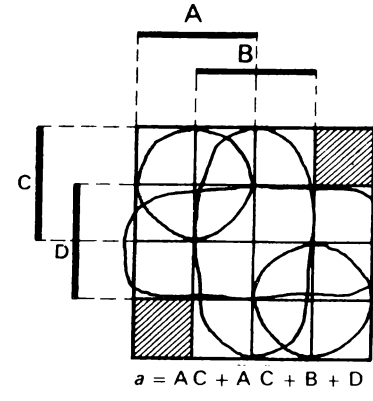


Fig. 4. 8

en nous référant à ce que nous avons vu dans le chapitre 2, nous savons que $AC + \bar{A}\bar{C}$ correspond au complément de $A\bar{C} + \bar{A}C$ qui est le OU-EXCLUSIF

$$a = \overline{A\bar{C} + \bar{A}C} + B + D = \overline{A \oplus C} + B + D$$

et si nous voulons utiliser des portes NAND, nous pouvons modifier encore cette expression :

$$a = \overline{\overline{A \oplus C} + \bar{B}\bar{D}} = (A \oplus C)BD$$

Le même travail de mise en correspondance avec les valeurs de ABCD et $\bar{A}B\bar{C}\bar{D}$ doit se faire pour chaque segment, sur le même principe. Il n'y aura plus alors qu'à passer à la réalisation en utilisant les portes correspondant aux relations établies.

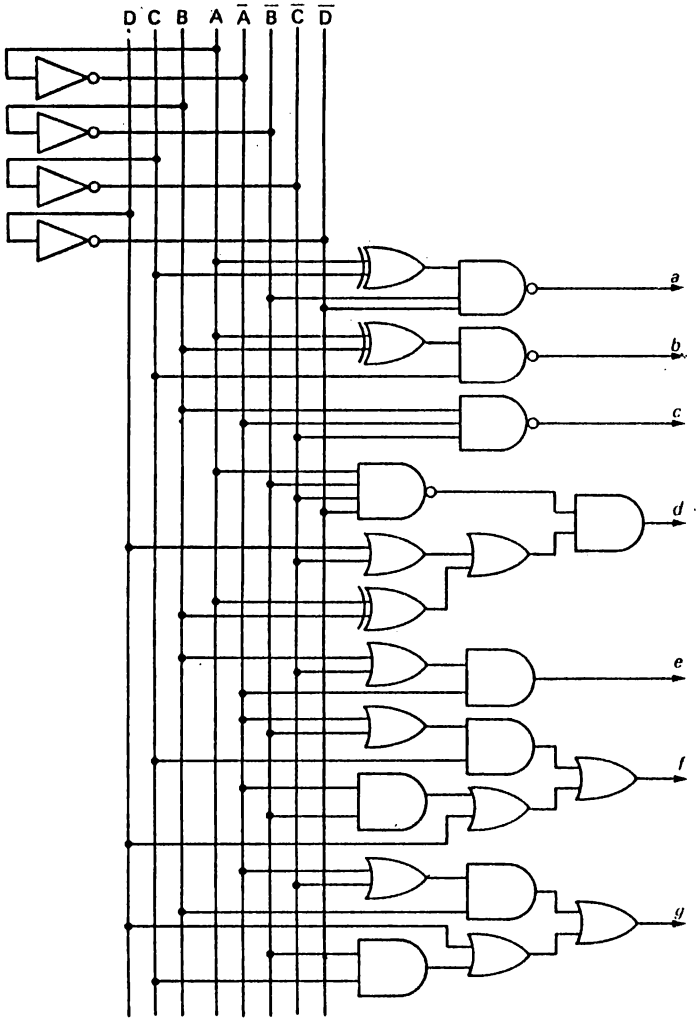


Fig. 4. 9

C'est ainsi qu'avec :

4 inverseurs	(1 boîtier 04)
1 NAND à 4 entrées	(1 boîtier 20)
2 NAND à 3 entrées 1 NAND à 2 entrées	(1 boîtier 10)
6 ET à 2 entrées	(2 boîtiers 08)
9 OU à 2 entrées	(3 boîtiers 32)
3 EXOR	(1 boîtier 86)

et un peu de patience, vous pourrez constituer le circuit de décodage de la figure 4. 9.

2.3.2 Décodeurs en circuits intégrés

Ce montage n'aura qu'un aspect pédagogique. Il existe en effet des décodeurs en circuits intégrés, basés sur le même principe, et qui présentent en outre quelques commandes supplémentaires.

Dans la série 74 nous trouvons les circuits 46 - 47 - 48 et 49 qui affichent des 6 et des 9 sans « queue » et les circuits 246 - 247 - 248 - 249 dont l'affichage est identique à celui que nous venons d'étudier.

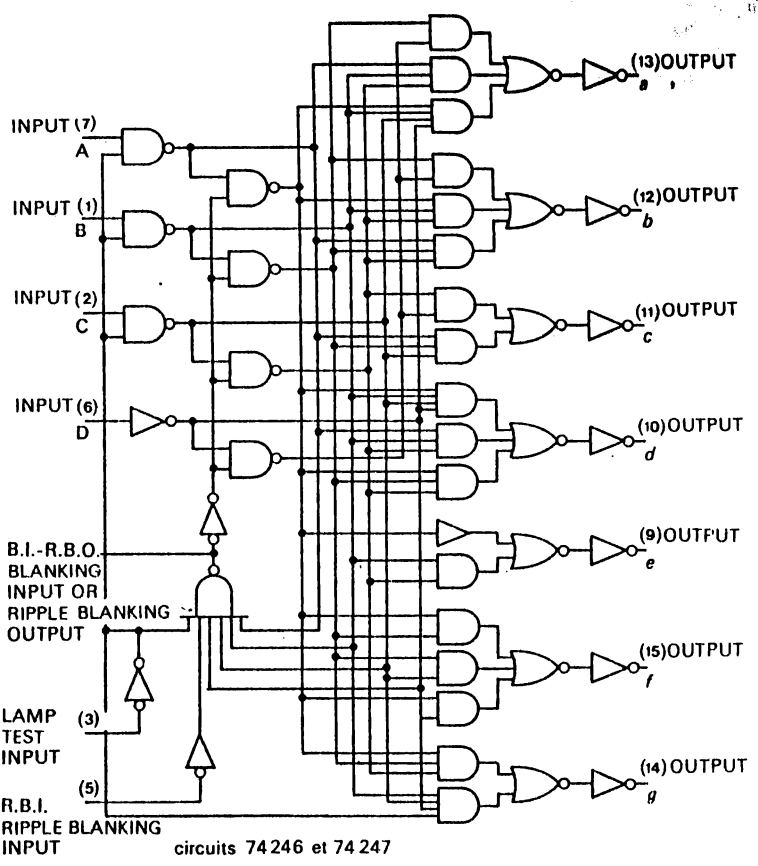


Fig. 4. 10 (Doc. Texas Instruments)

La figure 4. 10 montre le schéma interne des circuits 246 et 247. Les portes utilisées pour donner «*a*» ne sont pas les mêmes que celles adoptées sur le schéma 4. 9 mais le résultat est le même.

Nous y trouvons en outre une commande «lampe-test» qui permet de vérifier, en la mettant au niveau bas, que toutes les diodes fonctionnent correctement, ainsi qu'une commande «Blanking Input» (BI) qui permet de supprimer tout affichage au passage du zéro si l'on met cette commande au niveau bas.

Les afficheurs 7 segments proposés par les constructeurs sont de deux types : à anode commune ou à cathode commune. S'agissant de les employer avec un décodeur, le plus pratique est d'adopter le décodeur à collecteur ouvert et de brancher les segments sur le collecteur avec une résistance de protection. Les décodeurs 46, 47, 49 et 246, 247 et 249 répondent à ces besoins. Leur collecteur peut être branché à une source de 15 volts (247) ou même 30 volts (246). On n'oubliera pas de calculer la résistance de protection en fonction de cette tension, en tenant compte du fait qu'en fonctionnement normal une LED absorbe 20 mA et crée une chute de tension de 1,6 V.

Pour l'utilisation sous 5 volts cette résistance est habituellement de 150 à 220 ohms.

Chaque fois que l'homme et la machine ou deux machines ont à «dialoguer» alors qu'elles n'utilisent pas le même langage, un décodeur est nécessaire. Le décodeur BCD 7 segments n'est qu'un exemple particulier. Le décodeur proposé en exercice en est un autre exemple. La famille des décodeurs électroniques est très riche de modèles qui portent des noms très variés. Ils vont jouer un rôle très important dans les chapitres suivants de cet ouvrage.

3. LE CODE INTERNE DES MACHINES ÉLECTRONIQUES

Étant donnée la complexité des calculs en BCD, les nombres seront plutôt traités en code binaire pur à l'intérieur des machines électroniques.

A la simplicité de cette solution s'ajoute une économie de registres : le code BCD, en effet, n'utilise pas toutes les possibilités de combinaisons de quatre valeurs binaires puisque les valeurs correspondant aux nombres décimaux 10, 11, 12, 13, 14 et 15 ne sont pas utilisés (*fig.* 4. 11 *a*).

Tout se passe comme si, la machine ayant 16 doigts nous n'en utilisions que 10 sous prétexte que nous n'avons que 10.

3.1 Organisation des circuits en binaire pur

Rendons à la machine ses 16 « doigts ». Ainsi, avec un registre de 4 bits il sera possible de compter de 0 à 15. Pour aller plus loin il faudra un bit de plus, puis un second pour dépasser 31, un troisième pour dépasser 63, etc... Où s'arrêter?

Il a été convenu, pour la plupart des appareils électroniques, de constituer des registres unitaires de 4 bits, de 8 bits ou de 16 bits. L'élément le plus répandu étant l'octet, c'est-à-dire le groupe de 8 bits.

Un octet permet de compter, en binaire pur, jusqu'à 11111111, ce qui représente, en décimal, le nombre 255, c'est-à-dire le nombre $2^8 - 1$. De 0 jusqu'à 255 il y a bien 256 (2^8) nombres.

Nous verrons plus tard, en étudiant les micro-ordinateurs, comment on peut manipuler des nombres beaucoup plus grands, mais on peut aussi remarquer tout de suite que les ensembles électroniques dotés de registres à 16 bits pourront manipuler des nombres jusqu'à $2^{16} - 1$, c'est-à-dire jusqu'à 65535.

3.2 Le code hexadécimal

Comme nous avons pu déjà le constater, la lecture d'un nombre binaire pur n'est pas chose aisée et il est pratiquement impossible d'apprendre par cœur toutes les configurations possibles de 8 bits.

Déjà, pour 4 bits, l'exercice n'est pas toujours facile. L'homme, voulant toujours se faciliter la tâche, a donc décidé de transposer le code des 16 premiers chiffres (de 0 à 15) dans un langage plus classique pour lui.

Pour les 10 premiers chiffres, de 0 à 1001 en binaire, les équivalents existent : se sont les chiffres de 0 à 9.

Pour les 6 suivants il fallait inventer des symboles. Ceux-ci ont été puisés dans l'alphabet, en utilisant les lettres de A à F. Ainsi a été composé le code hexadécimal qui n'est qu'une transcription du binaire pur dans un code plus pratique décomposant ce code binaire en groupe de 4 bits (*fig.* 4. 11 *b*).

Ainsi le nombre binaire 01001010 se traduira-t-il en hexadécimal par 4 A, puisque le groupe de 4 bits de gauche représente le chiffre hexadécimal 4 et le groupe de droite le chiffre hexadécimal A (10 en décimal).

Décimal	Binaire (4 bits)			
	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

a

Décimal	Hexadécimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

b

Fig. 4. 11

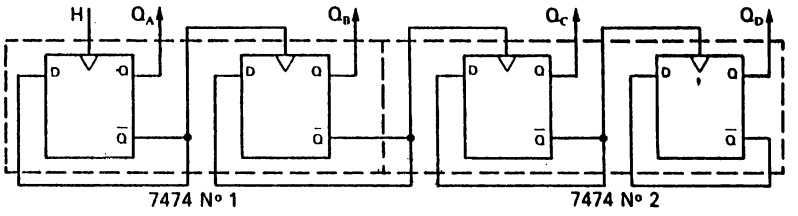


Fig. 4. 12

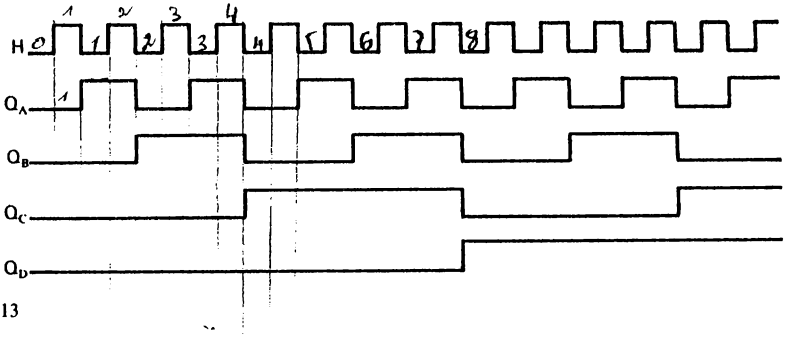


Fig. 4. 13

La traduction, en décimal, de ce nombre hexadécimal, se fait toujours suivant le principe des puissances successives de la base (ici 16)

$$4\ A_{16} = (4 \times 16^1 + 10)_{\text{base } 10}$$
$$= 64 + 10 = 74$$

Un peu plus facile à manier que le code binaire pur, ce code hexadécimal a pour principale qualité d'être beaucoup plus économique en signes. Il évite l'affichage de grandes quantités de 0 et de 1 et divise par 4 le nombre de ces signes.

On retiendra essentiellement qu'il n'est qu'un relais entre l'homme et la machine.

4. LE COMPTAGE

Tout comme l'homme qui veut se lancer dans des calculs doit avant tout savoir compter, il est indispensable que les machines électroniques soient dotées de compteurs.

4.1 Les compteurs en binaire pur de type asynchrone

Examinons le projet du § 1 (*fig.* 4. 2), en réalisant un compteur à 4 bits grâce à 2 circuits 7474 contenant chacun 2 bascules D (*fig.* 4. 12).

Rappelons-nous que pour que ces bascules changent d'état à chaque créneau d'une horloge, il faut relier la sortie $\overline{Q}$ à l'entrée D et envoyer un ordre « clear », avant le début du comptage, sur toutes les bascules.

Ainsi toutes les sorties Q sont à 0, mais toutes les entrées D sont à 1, et dès qu'un front positif se présentera sur Ck, la sortie Q passera à 1. Pour éviter que ce front positif de Q ne fasse changer d'état en même temps toutes les autres bascules, il faut relier l'entrée Ck des bascules suivantes à l'entrée $\overline{Q}$ de la précédente. C'est donc lorsque Q repassera à 0 (et $\overline{Q}$ à 1) que la bascule suivante changera d'état. Nous aurons alors le chronogramme de la figure 4. 13.

On peut imaginer de raccorder de la même façon autant de bascules que l'on veut. Seul défaut : chaque bascule étant commandée par la précédente, les changements d'état se font avec un délai qui, entre la première et la dernière bascule, risque de devenir significatif (environ 20 ns par bascule).

On dit que ces compteurs sont asynchrones, car, comme nous venons de le voir, les changements d'état ne se font pas tous ensemble. Ceci n'empêche pas ces compteurs de jouer un rôle utile dans différents cas, et surtout :

— lorsque la fréquence de l'horloge de base n'est pas trop élevée (au-dessous du mégahertz),

— lorsqu'il ne s'agit (cas analogue) que de compter des impulsions arrivant par moment (sans rythme précis),

— pour constituer des diviseurs par 4, 8, 16 etc... (utilisation dans les fréquencemètres).

4.2 Compteurs asynchrones en BCD

4.2.1 Modèle simple

Pour obtenir, sur le même principe, un compteur BCD, il faut revenir à 0 après la configuration du 9 (1 0 0 1). Un moyen très simple consiste à organiser cette remise à 0 par des portes, en utilisant des bascules munies de la commande Clear.

Ceci est encore possible avec les circuits intégrés 7474 en utilisant le schéma de la figure 4. 14 : l'état 1 0 1 0 étant à éviter, dès qu'il apparaît, la porte NAND provoque un créneau négatif sur les «Clear» des bascules et le comptage repart à 0.

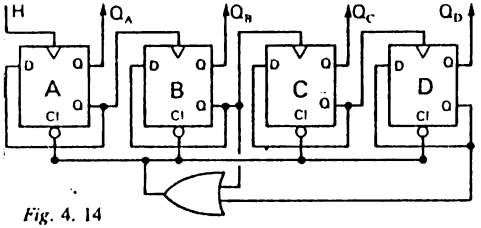


Fig. 4. 14

Ce schéma est, là encore, d'une grande simplicité. Il peut être généralisé à tout comptage partiel (0 à 3, 0 à 4, 0 à 7 par exemple) et pourra, à ce titre, servir aussi de diviseur de fréquence.

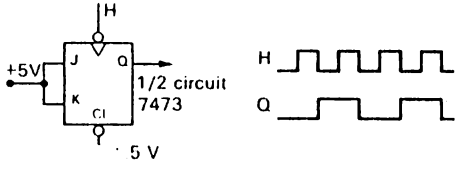
Mais il a aussi une faille : il faut que l'état que l'on veut éviter apparaisse, même très fugitivement, pour que la remise à 0 se fasse. Ceci n'est pas gênant aux basses fréquences mais peut devenir très dangereux aux hautes fréquences.

4.2.2 Modèle plus élaboré

Pour cette raison on a cherché à construire des compteurs asynchrones dont la remise à 0 soit préparée à l'avance pour intervenir au moment même d'une impulsion d'horloge.

On utilise alors des bascules JK en jouant par exemple sur la commande J.

Prenons le cas de la bascule JK 7473. Si sa sortie Q est à 0 et les deux commandes J et K à 1, la sortie $\overline{Q}$ passera à 1 après un créneau positif complet de l'horloge, au moment du front descendant (*fig.* 4. 15). Au second créneau de l'horloge Q retournera à 0, etc... (c'est ce que la documentation en anglais appelle «TOGGLE»).



Observez bien le chronogramme de la figure 4. 16. Un créneau positif sur J «prépositionne» la bascule pour un changement d'état qui se produira à l'impulsion suivante de l'horloge.

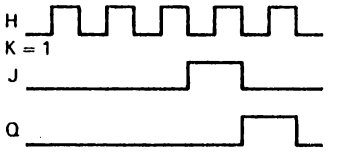


Fig. 4. 16

C'est cette propriété qui est exploitée dans la conception du compteur BCD de la figure 4. 17.

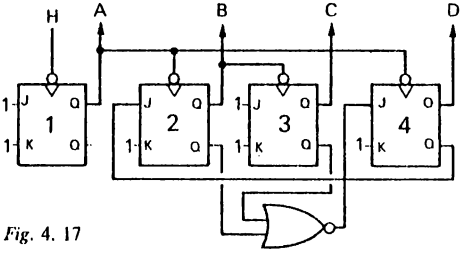


Fig. 4. 17

Deux prépositionnements y sont effectués : celui de la bascule 4 lorsque $Q_B \cdot Q_C = 1$, c'est-à-dire pour les nombres 1 1 0 et 1 1 1 (6 et 7); et celui de la bascule 2, lorsque $Q_4 = 1$, pour les nombres 8 et 9, obligeant cette bascule 2 à rester dans la position Q = 0 pour le début du nouveau cycle (0 et 1).

On notera qu'avec les bascules 7473 il est indispensable de relier les commandes non utilisées à l'alimentation (J, K ou Clear).

4.3 Compteurs asynchrones «modulo N»

«Compter jusqu'à N» est une expression ambiguë. S'agit-il de compter de 0 à N ou de 0 à N − 1.

C'est pourquoi les électroniciens ont décidé d'introduire un vocabulaire plus précis :

Un compteur «modulo 9» est un compteur de 0 à 8, qui compte donc 9 coups d'horloge, de même qu'un compteur modulo 5 est un compteur de 0 à 4.

Le compteur BCD à 4 bascules est un compteur modulo 10, celui en binaire pur est un compteur modulo 16, etc...

On pourra aussi parler d'un compteur à N positions pour dire la même chose.

Un compteur BCD qui compte de 0 à 9 est un compteur à 10 positions ou un compteur modulo 10.

Les fabricants de circuits intégrés proposent des compteurs asynchrones adaptables à différents cas. Ce sont les circuits 93, 90 et 92.

4.3.1 Le circuit 7493

C'est un compteur binaire constitué de 4 bascules JK. Il possède 4 commandes, 2 horloges et 2 commandes de remise à zéro (*fig.* 4. 18).

Si l'on n'utilise que l'entrée A d'horloge et la sortie Q_A on aura un compteur modulo 2.

En utilisant l'entrée d'horloge B et les sorties Q_B Q_C Q_D on a un compteur modulo 8.

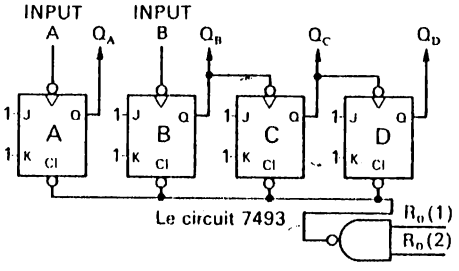


Fig. 4. 18

Enfin, en reliant Q_A à l'entrée d'horloge B on obtient le compteur binaire à 16 positions (modulo 16).

Les deux commandes R_0 portées au niveau 1 simultanément permettent de positionner toutes les sorties à 0.

4.3.2 Le circuit 7490

Ce circuit comporte 4 bascules. Les 3 premières sont des bascules JK et la dernière une bascule RS commandée directement par l'entrée d'horloge B. Les commandes R_0 toutes deux à 1 (avec au moins une commande R_9 à 0) entraînent la mise à 0 de toutes les bascules. Les commandes R_9 toutes deux à 1 entraînent la position 1001 (chiffre 9) (voir *fig.* 4. 19). Si au moins une des commandes R_0 et une des commandes R_9 est à 0, le circuit est en mesure de compter.

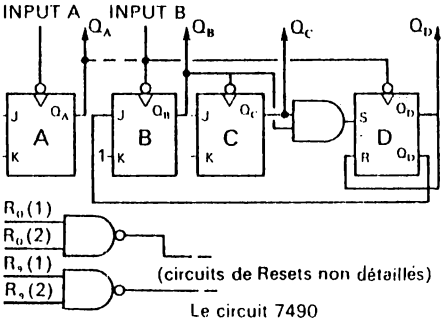


Fig. 4. 19

4.3.2.1 Compteur BCD

Supposons l'état initial 0000 et l'entrée B reliée à Q_A (trait en pointillés), et étudions le chronogramme (*fig.* 4.-20).

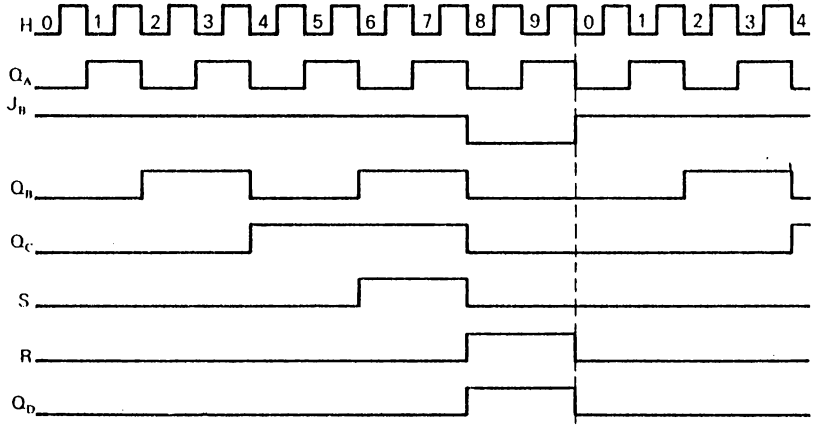


Fig. 4. 20

$\overline{Q_D}$ et donc J_B sont à 1, R et S sont à 0. Les bascules A, B et C vont fonctionner comme dans un compteur binaire jusqu'à 6. A ce moment-là Q_B et Q_C étant à 1 mettent l'entrée S à 1, prépositionnant le changement d'état de la bascule D. Ce changement n'aura lieu qu'à la fin du créneau d'horloge de Q_A correspondant à la position 7.

A l'état 8, Q_D passe à 1, $\overline{Q_D}$ à 0 ainsi que J_B. Le Reset de D passe à 1. Ceci dure pendant un créneau complet de A (états 8 et 9). A la fin de ce créneau (état 10) la bascule B est forcée à 0 par J_B, laissant Q_C dans son état (0) et Q_D est forcée à 0 par R. Le comptage recommence à 0000.

4.3.2.2 Le comptage « bi-quinaire »

La bascule A du circuit 7490 étant indépendante des autres peut être branchée non plus en tête mais en queue des trois autres, en reliant Q_D à Input A (*fig.* 4. 21).

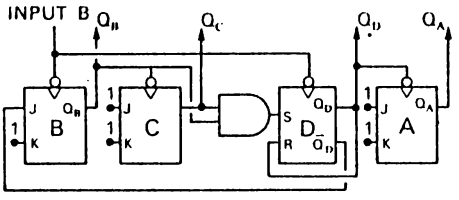


Fig. 4. 21

Il faut alors lire les sorties dans l'ordre de poids décroissant Q_A Q_D Q_C Q_B.

Une analyse du fonctionnement (guidez-vous sur le raisonnement précédent) donne le chronogramme et la table de vérité de la figure 4. 22.

S'il s'agit d'un nouveau code binaire, celui-ci a un intérêt essentiel, celui de générer en Q_A des créneaux symétriques qui divisent exactement par 5 la fréquence du signal qui était produit précédemment par la seule bascule Q_A. Il divise en fait par 10 la fréquence de l'horloge.

4.3.3 Le circuit 7492

Ce circuit crée, lui aussi, un code binaire particulier dont le principal intérêt est, cette fois, de fournir sur la sortie Q_D un signal dont la fréquence est le sixième de la fréquence de Q_A, et le douzième de la fréquence de l'horloge (input A).

Pour cela il faut brancher Q_A sur l'entrée B (voir *fig.* 4. 23 et 4. 24).

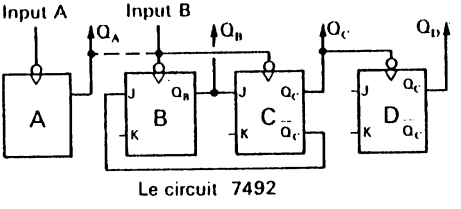


Fig. 4. 23

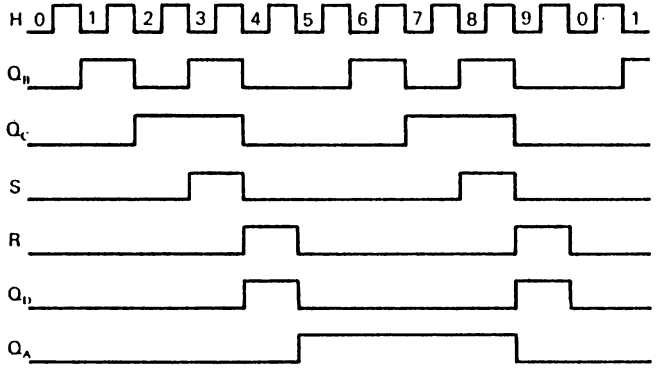
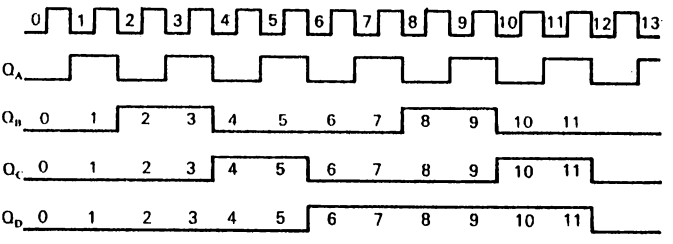


Fig. 4. 22

	Q_A	Q_D	Q_C	Q_B
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0



	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	1	0	0	0
7	1	0	0	1
8	1	0	1	0
9	1	0	1	1
10	1	1	0	0
11	1	1	0	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
etc...				

Fig. 4. 24

4.4 Les compteurs synchrones

Il est souvent nécessaire de synchroniser un ensemble d'opérations de traitement d'informations, c'est-à-dire de les référer à une horloge unique.

C'est pour cette raison qu'ont été développés les compteurs synchrones dont les types sont très nombreux.

Le composant le plus fréquent de ces compteurs et la bascule JK.

4.4.1 Compteur binaire synchrone

Étudions une unité de comptage synchrone à 4 bits pouvant compter de 0 à 15.

Une seule horloge H attaque l'entrée Ck de chaque bascule.

Quelles relations faut-il établir entre une sortie Q_N et les commandes JK de la bascule N + 1?

Pour répondre à cette question on peut établir le tableau de la figure 4. 25 où, pour chaque état, nous avons noté les basculements qui doivent survenir au coup d'horloge suivant.

Pour que le basculement puisse avoir lieu, il faut que dans la période précédente J et K soient à 1 pour la bascule concernée.

Pour la bascule A qui change d'état à chaque coup d'horloge, J et K seront en permanence à 1.

Pour la bascule B, J et K doivent être à 1 lorsque Q_A est à 1. On les branchera donc sur Q_A.

La bascule C ne doit basculer que lorsque Q_A et Q_B sont à 1. Il faut ajouter une porte AND sur l'entrée de J_C avec les entrées sur cette porte de Q_A et de Q_B.

État	Q_D	Q_C	Q_B	Q_A	Prévoir le basculement de :
0	0	0	0	0	Q_A
1	0	0	0	1	Q_A Q_B
2	0	0	1	0	Q_A
3	0	0	1	1	Q_A Q_B Q_C
4	0	1	0	0	Q_A
5	0	1	0	1	Q_A Q_B
6	0	1	1	0	Q_A
7	0	1	1	1	Q_A Q_B Q_C Q_D
8	1	0	0	0	Q_A
9	1	0	0	1	Q_A Q_B
10	1	0	1	0	Q_A
11	1	0	1	1	Q_A Q_B Q_C
12	1	1	0	0	Q_A
13	1	1	0	1	Q_A Q_B
14	1	1	1	0	Q_A
15	1	1	1	1	Q_A Q_B Q_C Q_D

Fig. 4. 25

Enfin, comme Q_D ne doit basculer que lorsque Q_A, Q_B et Q_C sont à 1, on ajoutera une nouvelle porte AND sur J_D.

On obtient alors le compteur synchrone modulo 16 de la figure 4. 26.

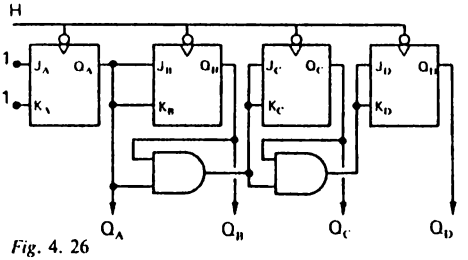


Fig. 4. 26

4.4.2 Étages supplémentaires

Si l'on veut pouvoir compter jusqu'à 255 ou bien au-delà il faut pouvoir «raccrocher» d'autres blocs de 4 bits à ce premier compteur. Mais il ne suffira plus de brancher la sortie Q_D du premier compteur sur l'entrée Q'_A du deuxième. Il faudra songer aussi à brancher sur les commandes J et K de cette bascule A' les fils qui la feront basculer au bon moment.

Si l'on prolonge le raisonnement fait précédemment, il faudra, à l'entrée de A', une porte AND qui reçoive les valeurs de J_DK_D et de Q_D (*fig.* 4. 27).

4.4.3 Les problèmes liés aux délais de propagation

On remarquera alors que les commandes J et K de la bascule D' reçoivent alors l'information Q_A à travers 6 portes en série. Pour les électroniciens ceci signifie qu'il faudra multiplier par six le temps de propagation de la donnée Q_A à travers chaque porte. Pour de hautes fréquences ceci constitue à nouveau un handicap. En fait la fréquence d'utilisation limite sera d'autant plus basse qu'on

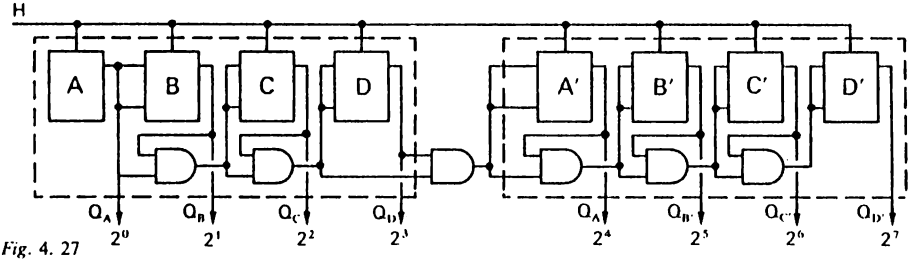


Fig. 4. 27

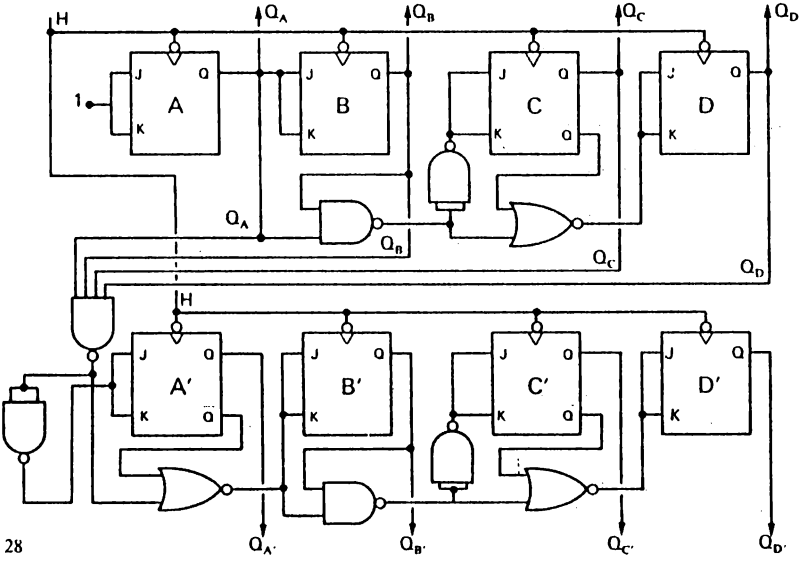


Fig. 4. 28

utilisera davantage de registres raccordés les uns aux autres.

Deux solutions peuvent être adoptées pour améliorer ce fonctionnement :

a) remplacer les portes AND par des portes inverseuses qui sont plus rapides (pour inverser il suffit d'un transistor). Par exemple le délai de propagation dans le cas où une sortie de porte passe de 0 à 1 est de l'ordre de 18 ns pour la porte 7400 et de l'ordre de 50 ns pour la porte 7401,

b) reprendre directement les valeurs des sorties de Q_A à Q_D pour le deuxième groupe de bascules.

Ces deux solutions sont mises en application simultanément dans le schéma de la figure 4. 28. Le type de compteur vu précédemment est qualifié de compteur synchrone «série» tandis que celui-ci est un compteur synchrone «parallèle». Il est capable de répondre à des fréquences d'horloge de l'ordre de 30 MHz.

4.4.4 Compteurs BCD synchrones

Il est possible de réaliser un compteur BCD synchrone par des modifications légères du compteur binaire.

Rappelons les conditions de fonctionnement de la bascule JK du type 7473 :

Si J = 1 et K = 1 : basculement au top d'horloge suivant.

Si J = 1 et K = 0 : Q passe à 1 (ou y reste) au top suivant.

Si J = 0 et K = 1 : Q passe à 0 (ou y reste) au top suivant.

Si J = 0 et K = 0 : Q et $\overline{Q}$ restent en l'état.

Il faut que le compteur retombe à 0 après l'état 9 (1001).

Le compteur A continue à basculer à chaque coup d'horloge, aucune modification n'est nécessaire.

Q_B doit rester à 0 soit en faisant $J_B = 0$ et $K_B = 1$, soit $J_B = 0$ et $K_B = 0$.

Dans le compteur binaire nous avions $J_B = K_B = Q_A$. Il faut encore que Q_B soit à 0 après 1000 et 1001, soit $J_B = K_B = \overline{Q_D}$. On fera donc $J_B = K_B = Q_A\overline{Q_D}$.

J_C et K_C sont liés à $Q_A \cdot Q_B$. Il n'y a pas de problème à ce niveau-là.

Q_D devant passer à 0, la condition $J_D = K_D = Q_AQ_BQ_C$ n'est pas suffisante (dans ce cas J_D et K_D sont égaux à 0 pour la position 9 et Q_D ne basculerait pas). On peut forcer Q_D à 0 en faisant apparaître un 1 sur K_D par $K_D = Q_A$.

$J_D = Q_AQ_BQ_C$ (inchangé) et $K_D = Q_A$ répondent au problème. A la position 7 (ou ABC sont à 1) on aura $J_D = 1$ et $K_D = 1$ et la bascule D changera d'état. A la position 9, on aura $J_D = 0$ et $K_D = 1$ et la bascule D sera forcée à 0.

La réalisation du compteur BCD à 4 registres est alors représentée par la figure 4. 29.

4.4.5 Étages supplémentaires

Le compteur précédent à 10 positions (modulo 10) est aussi appelé une «décade». En mettant bout à bout plusieurs décades on aura un compteur BCD transposant en binaire la structure du code décimal : chaque bloc de 4 compteurs représentant un chiffre décimal de rang 1, 2, 3 etc...

Pour réaliser ce compteur il faut relier les étages de rang supérieur au premier étage par des portes permettant de ne comptabiliser que les dépassements : chaque fois qu'un étage aura enregistré 1001 il transmettra un créneau d'horloge à l'étage suivant.

En utilisant les portes NAND et NOR on obtient alors le schéma de la figure 4. 30 (nous n'avons pas représenté les 4 bascules pour simplifier le schéma).

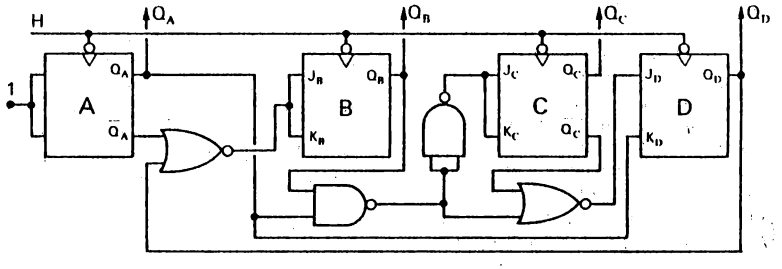


Fig. 4. 29

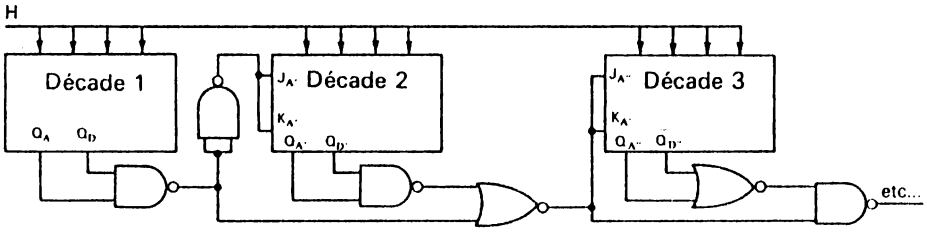


Fig. 4. 30

4.5 Circuits intégrés

Les compteurs synchrones sont proposés en circuits intégrés sous un grand nombre de références : 74160, 161, 162, 163, 190, 191, 192, 193 etc...

Généralement ils sont à la fois compteurs et « décompteurs ». De plus ils peuvent souvent être prépositionnés sur une valeur donnée.

Nous allons examiner successivement ces deux améliorations.

4.5.1 Compteurs-décompteurs

Si on observe la figure 4. 31 on voit qu'il y a une seule différence entre compter de 0 à 15 et décompter de 15 à 0 : il suffit de changer les 0 en 1 et les 1 en 0!

Comptage					Décomptage				
0	0	0	0	0	1	1	1	1	15
0	0	0	1	1	1	1	1	0	14
0	0	1	0	2	1	1	0	1	13
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	1	0	1	1	11
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	1	0	0	1	9
0	1	1	1	7	1	0	0	0	8
1	0	0	0	8	0	1	1	1	7
1	0	0	1	9	0	1	1	0	6
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	0	1	0	0	4
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	0	0	1	0	2
1	1	1	0	14	0	0	0	1	1
1	1	1	1	15	0	0	0	0	0

Fig. 4. 31

Dans un « décomptage » la bascule N doit donc changer d'état si la sortie de la bascule précédente est à 0. Il faut donc établir un nouveau réseau logique en utilisant, en commande d'entrée de chaque porte les valeurs complémentaires de celles utilisées pour compter.

Mieux : les deux réseaux logiques peuvent être présents sur le même circuit compteur pourvu qu'ils n'agissent pas en même temps. Il suffira pour cela de mettre une autorisation aux portes qui doivent être utilisées (une commande supplémentaire à 1 sur un AND par exemple autorise cette porte. Si elle est à 0 elle la condamne).

La construction de compteurs-décompteurs en BCD, pour moins évidente qu'elle soit, est également réalisable.

C'est ainsi que les constructeurs proposent essentiellement deux types de circuits :

— les 74168 (décade BCD) ou 169 (binaire) disposant d'une seule horloge. Ces compteurs comptent si le signal U/$\overline{D}$ (up/down) est haut, et décomptent si U/$\overline{D}$ est bas,

— les 192 (décade BCD) ou 193 (binaire) disposant de deux entrées de bascule, l'une pour compter et l'autre pour décompter.

4.5.2 Utilisation des compteurs-décompteurs

A chacun de ces circuits correspondent des utilisations déterminées :

— Pour un chronomètre permettant de compter le temps, puis de le décompter à partir d'un moment donné, il n'y aura qu'une entrée d'horloge et un signal compte/décompte. C'est le circuit 168 qui s'impose.

— Pour connaître le nombre de personnes présentes dans un musée par exemple on peut compter celles qui entrent par le portillon d'entrée et décompter simultanément celles qui sortent par le portillon de sortie. Il y aura deux horloges. C'est le circuit 192 qui s'impose.

4.5.3 Prépositionnement à une valeur donnée

Ce prépositionnement sert à imposer une valeur déterminée aux sorties, par exemple lorsque l'on veut compter à partir de 1 et non de 0.

Une entrée « chargement » permet de forcer les entrées J et K de telle façon que les sorties soient égales aux consignes (valeurs à prépositionner).

Lorsque cette entrée « chargement » (« Load » en anglais) est activée les entrées J recopient les valeurs à imposer et les entrées K leurs

compléments. Après désactivation de « Load » le compte commencera sur la valeur imposée.

Un tel compteur est dit « programmable » tout comme un four « programmable » qui se met en route à l'heure indiquée par la cuisinière.

Les compteurs 74162-163 et 74192-193 sont programmables aussi bien pour compter que pour décompter.

Les mêmes compteurs peuvent être raccordés les uns aux autres suivant le plan de la figure 4. 30 et disposent, pour cela d'une sortie « Ripple Carry Out » (sortie d'un créneau de retenue) qui envoie une impulsion chaque fois qu'il y a débordement (le 9 étant atteint pour le BCD, le 15 pour le binaire).

4.6 Compteur de Johnson

Au chapitre précédent nous avons proposé, en exercice, la réalisation d'un chenillard utilisant des bascules branchées en série, la sortie Q de l'une sur l'entrée D de la suivante.

C'est sur ce principe qu'est construit le compteur de Johnson.

4.6.1 Code utilisé

En utilisant le principe du chenillard, une seule des sorties Q est à 1, toutes les autres sont à 0. Au coup d'horloge suivant c'est la bascule suivante qui passe à 1, la précédente retombant à 0. Avec 10 bascules on pourra donc constituer un compteur modulo 10 dont chaque sortie de bascule représentera un chiffre. Ce procédé n'est pas économique.

H	A	B	C	D	E
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1

Fig. 4. 32

En faisant circuler non plus un seul 1, mais une série de 1, on peut constituer un compteur modulo 10 avec 5 bascules. La table de vérité est représentée par la figure 4. 32.

En revanche le décodage sera un peu plus compliqué puisqu'il n'y a plus une sortie Q par chiffre.

Il faut décoder la transition entre les 0 et les 1, et une série de 10 portes à double entrée sera nécessaire. Vérifiez sur la figure 4. 33 comment fonctionne un tel décodage.

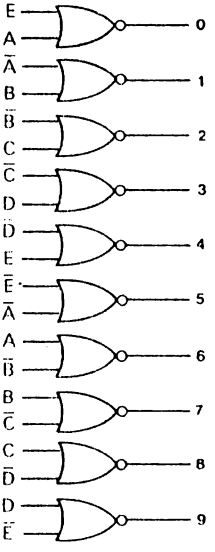


Fig. 4. 33

4.6.2 Réalisation

Ce compteur peut être réalisé avec des bascules D, disposant d'un « Clear ».

Une impulsion sur l'ensemble de ces remises à 0 permet de mettre le compteur « en phase » pour commencer le comptage.

Pour permettre alors l'apparition d'un 1 sur la sortie Q_A il faut brancher l'entrée D de cette bascule A sur la sortie $\overline{Q_E}$.

— Dès que le premier 1 aura atteint la bascule E (chiffre 5) cette sortie $\overline{Q_E}$ mettra l'entrée D_A à 0 et la sortie Q_A à 0 au coup d'horloge suivant.

Après remise à 0 on peut constater que le premier coup d'horloge met le compteur à 1. Si on voulait obtenir un 0 lors de ce premier coup, il faudrait positionner préalablement le compteur au chiffre 9 en agissant sur la mise à 0 des bascules A à D et sur la mise à 1 de la bascule E, ce qui est possible avec les circuits 7474.

On peut, du reste, en disposant à son gré des Preset et Clear, faire démarrer le comptage à une valeur quelconque, à condition qu'elle soit compatible avec le code utilisé.

5. APPLICATION : RÉALISATION D'UN FRÉQUENCEMÈTRE

Pour bien préciser le rôle que peuvent jouer les compteurs asynchrones ou synchrones, les diviseurs par N et les bascules elles-mêmes, le mieux est d'en faire une application pratique. Celle qui vous est proposée ici permet de passer en revue la plupart des circuits qui viennent d'être étudiés et de comprendre leur fonction particulière.

5.1 Le projet

Il s'agit de réaliser un outil de mesure de la fréquence d'un signal, en supposant cette fréquence stable. La forme elle-même du signal n'entre pas en jeu, mais si ce signal contient plusieurs fréquences, seule la fréquence dominante sera l'objet de la mesure. (Pour connaître les autres fréquences il faudra, au préalable, les filtrer.)

Il s'agit donc de compter le nombre de périodes du signal entré dans l'appareil en une seconde (ou un sous-multiple ou un multiple de la seconde). L'appareil devra donc comporter des compteurs et une horloge interne marquant les secondes.

5.2 L'horloge interne

Pour obtenir la précision du temps qui s'écoule, il faut faire appel au quartz. Ceux qui sont les plus accessibles sur le marché génèrent des signaux de plusieurs mégahertz.

La réalisation la plus pratique consiste à utiliser 1 quartz de 1 MHz et 2 inverseurs d'un circuit 74LS04. Un troisième inverseur servira à la mise en forme du signal sortant. Le montage est décrit sur la figure 4. 34.

Ce montage oscille spontanément pour une tension de l'ordre de 4 volts. L'alimentation étant à 5 volts, il est nécessaire de placer une résistance de 150 ohms pour obtenir à coup sûr ces oscillations.

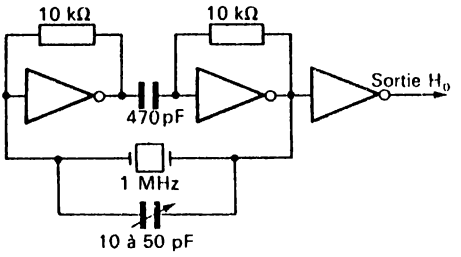


Fig. 4. 34

Le condensateur variable placé en parallèle avec le quartz permettra un ajustage fin de la fréquence (étalonnage).

Ayant obtenu une horloge qui bat le millionième de seconde, il faut ajouter des diviseurs de fréquence pour obtenir la seconde.

Ici peu importe le décalage possible entre les signaux de l'oscillateur et le signal qui servira de référence de temps. Nous pouvons utiliser des compteurs asynchrones. Le circuit 7490 (voir § 4.3.2.2) et son code bi-quinaire est très utile ici comme diviseur par 10.

En sortie du premier 90 nous aurons un signal carré de 100 kHz, en sortie du second, un signal de 10 kHz, puis 1 kHz, 100 Hz, 10 Hz et enfin 1 Hz (voir *fig.* 4. 35).

Suivant la précision que l'on veut obtenir, on utilisera l'une ou l'autre de ces horloges : une horloge à 1 kHz mesurera le nombre de périodes d'un signal pendant un millième de seconde et ne pourra pas indiquer les unités, dizaines et centaines.

C'est aussi un choix économique : pour mesurer une fréquence de 1.823.627 Hertz au Hertz près, il faudra 7 afficheurs.

A chacun de choisir sa solution. Supposons cependant que le choix se porte sur la sortie 1 Hz. C'est dorénavant ce signal qui nous servira de base de temps. Appelons-le H.

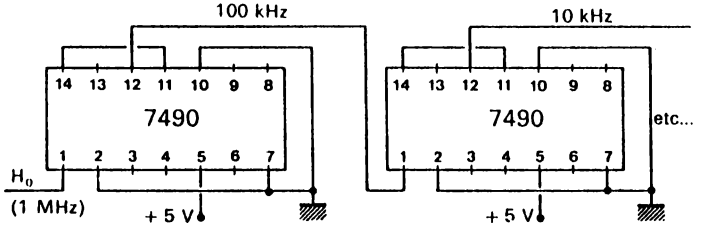


Fig. 4. 35

5.3 Entrée du signal à mesurer

Nous ne développerons pas ici le système qui permet la saisie du signal. Ce problème relève de l'électronique analogique. Il ne faut pas oublier que le signal d'entrée doit pouvoir être faible (quelques millivolts) ou important (quelques volts) et que sa fréquence peut aller de quelques hertz à quelques mégahertz.

Ne sous-estimons pas la difficulté de réaliser une telle entrée. Notons cependant que l'amplificateur d'entrée ne doit pas reproduire ce signal mais bien plutôt le transformer en créneaux : il faut donc que cet amplificateur fonctionne constamment en régime saturé ou bloqué.

5.4 Comptage des impulsions

Les impulsions obtenues à partir du signal d'entrée doivent maintenant être comptées.

L'aboutissement de ce comptage devant être un affichage 7 segments, nous adoptons d'emblée le code BCD.

Par ailleurs à partir de maintenant tout doit être synchrone : il ne s'agit plus de diviser des fréquences mais de compter pendant la base de temps d'une seconde que va nous fournir l'horloge H.

Faisons appel, pour ce comptage, au circuit 74 160 dont les caractéristiques figurent sur le tableau 4. 36.

L'horloge de ce circuit (plot 2) sera reliée au signal à mesurer. La durée du comptage sera définie par la durée d'un signal positif sur

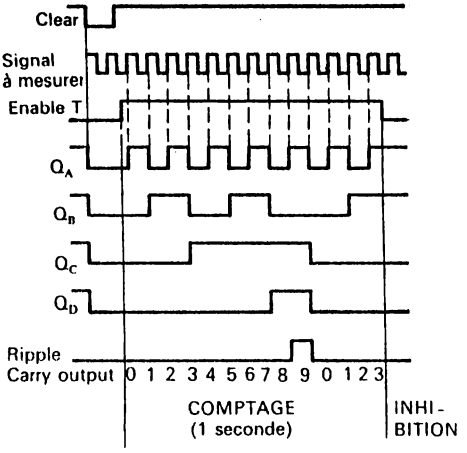


Fig. 4. 36

« Enable T ». Pour que cette durée soit d'une seconde, il faut que notre signal H passe par un diviseur par 2 (bascule D) pour fournir le signal H/2.

A la fin du créneau positif de H/2 le comptage sera inhibé pendant une seconde (durée du créneau négatif de H/2).

C'est pendant ce temps-là que le signal doit être lu et mis en mémoire pour être affiché.

5.5 Lecture du comptage (voir *fig.* 4. 37)

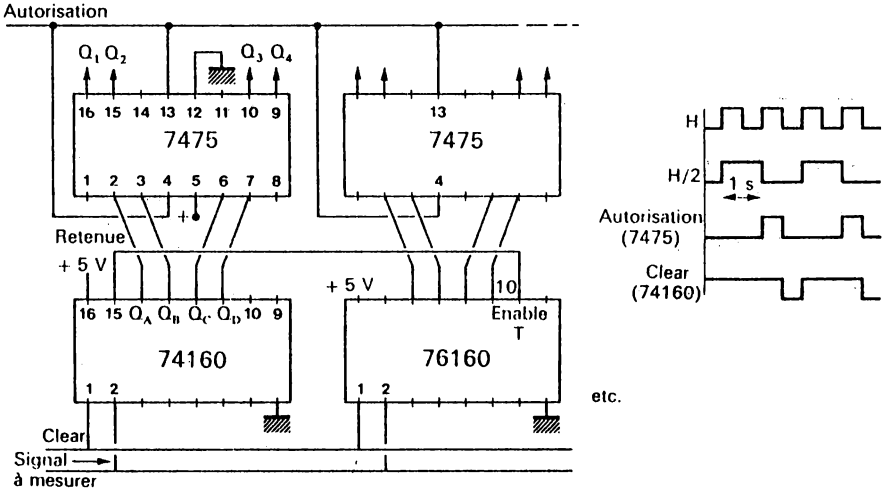


Fig. 4. 37

Les sorties 11 à 14 du circuit 160 délivrent les 4 bits DCBA du comptage BCD.

Utilisons une bascule D pour saisir chacune de ces valeurs. Le circuit 7475 possède 4 de ces bascules. Il est tout désigné pour remplir ce rôle.

Pendant la période d'inhibition du compteur 160, il faut non seulement lire les sorties, mais aussi mettre à 0 ce compteur pour recommencer l'opération de comptage.

La lecture doit donc avoir lieu pendant la demi-seconde qui suit le comptage.

On peut, à partir de H et de H/2 obtenir 1 créneau positif d'une demi-seconde qui, envoyé sur les plots 4 et 13 du 7475 (Enable 3-4 et Enable 1-2) permettront la lecture des valeurs fournies par le circuit 74 160, valeurs amenées sur les bornes 1D, 2D, 3D et 4D du 7475.

Ce signal correspond à $H \cdot \overline{H/2}$.

5.6 Remise à zéro du compteur

Cette remise à zéro se fait par un créneau négatif sur la borne « Clear » (N° 1) du circuit 160. On obtient ce créneau par la relation $H + H/2$ ou $\overline{H \cdot H/2}$.

5.7 Étages supplémentaires

Avec un seul compteur 160 on ne peut compter que jusqu'à 9, avec deux jusqu'à 99. Il en faudra donc de 4 à 8 pour satisfaire à différentes exigences.

Ces circuits peuvent être branchés en cascade et commandés par la même fréquence (signal à mesurer). Ainsi, dès qu'il y a dépassement, le bit supplémentaire est pris en compte par le compteur suivant.

Il faudra seulement s'assurer que s'il y a dépassement dans le compteur de poids le plus fort, ce dépassement sera signalé sur le tableau du fréquencemètre, par exemple, par l'allumage d'une LED.

Le montage en cascade s'effectue suivant le schéma, en branchant la sortie 15 d'un compteur (retenue) sur l'entrée Enable P et T du compteur suivant.

Les circuits 7475 associés devront être de même nombre que les circuits 74160.

5.8 Affichage de la mesure

Restera alors à prévoir le circuit d'affichage, c'est-à-dire un décodeur et un afficheur 7 segments pour chaque unité de comptage et de mise en mémoire.

Mais, dès le prochain chapitre, nous verrons qu'il y a une meilleure solution, c'est-à-dire une solution plus économique en circuits et en connexions grâce au multiplexage.

5.9 Aménagements conseillés

Avant de construire ce fréquencemètre sur des circuits imprimés, il est bon, comme pour tout montage, de le mettre à l'essai sur des plaques de montage rapide.

On y constatera quelques problèmes liés aux bruits parasites provoqués par les changements d'état. Quelques condensateurs bien placés peuvent améliorer cela.

Par ailleurs nous avons jugé bon, à l'expérience, de transformer le signal de lecture des compteurs 160 en le faisant passer à travers une bascule monostable type 74121 suivant le schéma 4. 38. Le signal, plus bref, est suffisant pour la lecture, et il est, de plus, nettoyé des bruits qui pourraient entraîner des basculements intempestifs des 7475. L'ajout d'un circuit 7416 à collecteur ouvert permet d'avoir la puissance nécessaire si les bascules 75 sont en nombre important.

Enfin si le signal à mesurer a une fréquence très élevée, de plusieurs mégahertz, mieux vaut le diviser par 10 (même principe que pour la base de temps) grâce à un commutateur, sachant que la lecture des afficheurs sera à multiplier par 10.

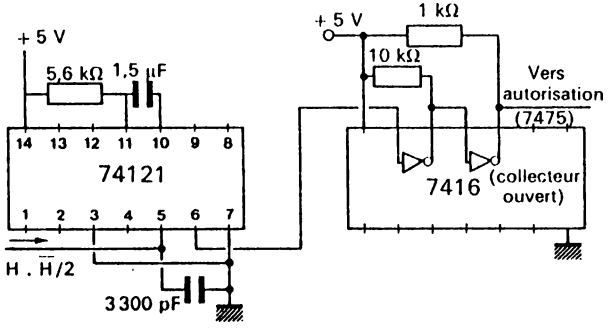
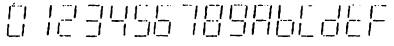


Fig. 4. 38

EXERCICES

On veut réaliser un décodeur binaire-sept segments, pour l'affichage, sur un afficheur à anode commune, des 16 valeurs d'un registre de 4 bits DCBA.

Les symboles utilisés pour l'afficheur doivent être les suivants :



Trouver les relations qui lient l'allumage de chaque segment aux valeurs de DCB et A.

On utilisera la méthode proposée au § 2.31 du présent chapitre, en étudiant les expressions $\bar{a}$, $\bar{b}$, $\bar{c}$ etc.

Imaginer un décodeur n'utilisant que des portes NAND.

CHAPITRE 5

Le multiplexage et les registres à décalage

1. PRINCIPE GÉNÉRAL DU MULTIPLEXAGE

Le multiplexage est l'opération qui consiste à faire circuler sur *un seul conducteur* des informations provenant de *sources différentes* et destinées à différents *récepteurs*.

Pour mieux comprendre cette notion, il suffit de se référer au principe du trafic ferroviaire : sur le conducteur unique qu'est une voie ferrée peuvent circuler successivement plusieurs trains qui viennent de lieux différents et vont vers des destinations différentes. Le système de « multiplexage » de ces trains est assuré d'une part par des aiguillages et, d'autre part, par un horaire.

Ces mêmes notions d'horaire et d'aiguillages sont les deux éléments de base du multiplexage électronique.

1.1 Multiplexage de 4 afficheurs 7 segments

A l'origine (gares de départs) 4 chiffres décimaux codés en B, C, D dans 4 circuits 7475 par exemple (chacun ayant 4 bascules) doivent être décodés pour être transformés en 4 fois 7 éléments binaires destinés aux 7 segments de chacun des 4 afficheurs.

Entre les 7475 et les décodeurs il faudra 16 fils.

Entre les décodeurs et les afficheurs il faudra 28 fils plus les 2 fils d'alimentation, soit 30 fils.

Nous allons voir comment économiser des circuits et des conducteurs grâce au multiplexage.

1.1.2 Un horaire

Appelons G_0, G_1, G_2 et G_3 les 4 gares de départ (les circuits 7475).

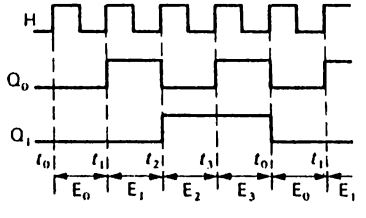
Chaque gare contient 4 éléments (trains) à faire circuler. Nous les appellerons A_0, B_0, C_0, D_0 pour la gare G_0, A_1, B_1, C_1 et D_1 pour G_1, etc.

Nous allons créer une voie commune à 4 « rails » et décider de 4 « époques » différentes, E_0, E_1, E_2 et E_3 :

– à l'époque E_0 ce sont les bits A_0, B_0, C_0, D_0 qui doivent circuler sur les conducteurs communs,

– à l'époque E_1, ce seront les bits A_1, B_1, C_1, D_1 etc.

Pour matérialiser ces 4 époques il nous faut un horaire officiel, facile à établir à partir du signal donné par un multivibrateur astable et à partir de 2 bascules branchées en compteur binaire (*fig.* 5. 1).



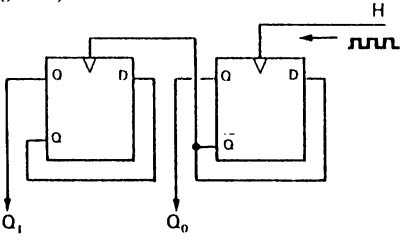


Fig. 5. 1

Cette horloge à 4 temps nous fournira donc, sur les sorties Q_1 et Q_0 les indications 00 pour le temps t_0, 01 pour le temps t_1, 10 pour le temps t_2 et 11 pour le temps t_3.

1.1.3 Des aiguillages

Il faut maintenant organiser les aiguillages pour que, de façon automatique, ce soit les valeurs A_N, B_N, C_N, D_N (N pouvant prendre les valeurs 0 à 3) qui circulent sur les conducteurs communs pendant l'époque E_N.

Pour réaliser ces aiguillages, il faut mettre les gares en communication avec les voies communes à travers des portes qui autoriseront ou non le passage des valeurs en fonction de l'horaire.

La figure 5. 2 montre la façon de procéder pour un seul des 4 bits A_N, B_N, C_N ou D_N.

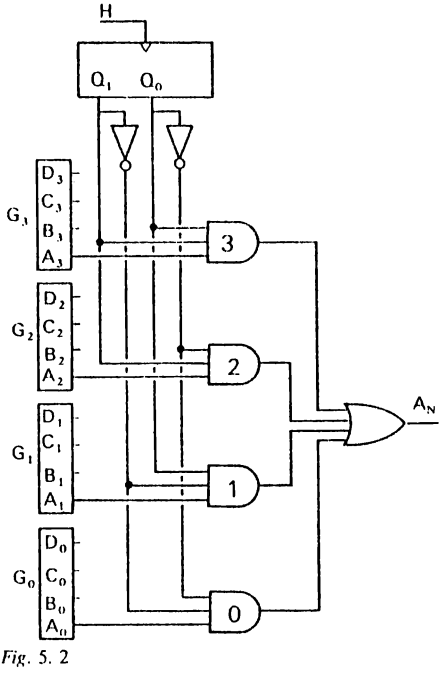


Fig. 5. 2

Grâce à 4 circuits AND et à 1 OR, la valeur A_N sera la seule à être « triée » à l'époque E_N.

Vérifier, sur ce schéma, qu'à chacune des époques E_N définie par le chronogramme de la figure 5. 1, c'est bien l'information A_N correspondante qui est sélectionnée.

Il faudra créer un dispositif identique pour les valeurs B (et un fil commun B_N) puis C et D. En tout nous aurons besoin de 20 portes.

L'ensemble de ces 20 portes qui relient les 4 circuits 7475 aux 4 fils de la voie commune et qui sont commandées par les 2 bits de l'horaire s'appelle un multiplexeur (*fig.* 5. 3).

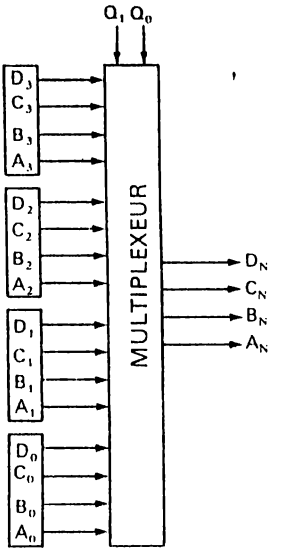


Fig. 5. 3

1.2 Le démultiplexage

A l'autre bout du circuit commun il faut organiser la redistribution des informations vers les 4 afficheurs.

Une astuce va permettre cependant d'économiser 3 des 4 décodeurs BCD. 7 segments qui paraissent d'emblée nécessaires.

Faisons donc aboutir les 4 conducteurs communs à un seul décodeur.

Les sorties de ce décodeur seront en communication avec les segments identiques correspondants de chaque afficheur. Par exemple la sortie « *a* » sera en communication avec les segments « *a* » des 4 afficheurs (*fig.* 5. 4).

A l'époque E_0, ce sont les données $a_0 b_0 c_0 d_0 e_0 f_0 g_0 h_0$ destinées à l'afficheur des unités que le décodeur fait parvenir à ces 4 afficheurs. Or seul l'afficheur des unités doit les recevoir. Il faut donc neutraliser les 3 autres afficheurs (D pour dizaines, C pour centaines et M pour milliers) pendant l'époque E_0.

Pour l'époque E_1, seul D devra être activé, etc.

Utilisons pour cela des afficheurs à cathode commune. Pour activer un tel afficheur, il suffit de relier cette cathode à la masse. Pour le désactiver il suffit de porter au potentiel + cette même cathode.

La solution consiste donc à relier la cathode de chaque afficheur, non pas à la masse, mais à un fil de commande qui sera porté au niveau bas pendant l'époque où le décodeur fait parvenir les informations qui lui sont destinées.

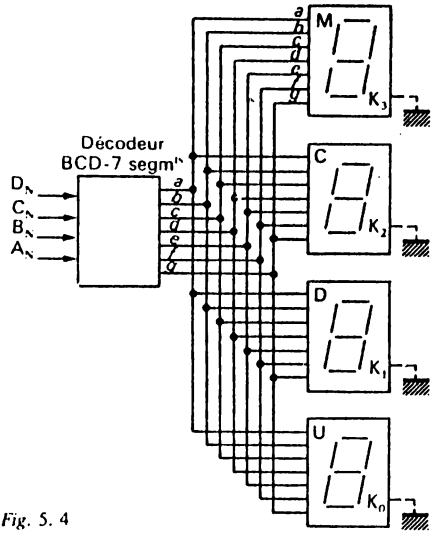


Fig. 5. 4

Le chronogramme de commande des 4 afficheurs est rigoureusement lié aux époques déterminées par l'horloge de multiplexage (*fig.* 5. 5).

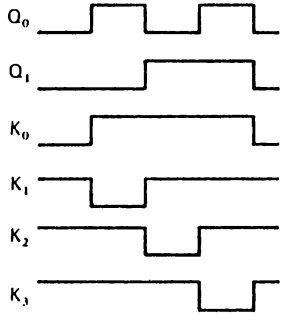


Fig. 5. 5

Pour obtenir les impulsions sur les fils de commande des cathodes, quelques portes suffiront.

On peut adopter par exemple le système représenté en figure 5. 6, à partir des sorties Q_0 et Q_1 des 2 bascules de l'horaire. L'ensemble de ces portes s'appelle un *démultiplexeur.*

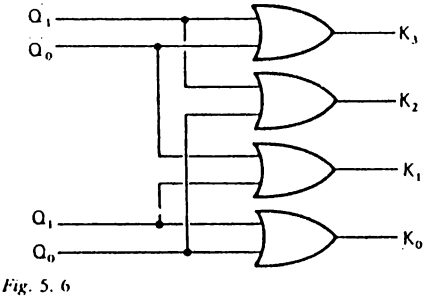


Fig. 5. 6

1.3 Résultats

On peut se demander si l'économie de quelques circuits et quelques fils ne se fait pas aux dépens d'un clignotement pénible des afficheurs. Faisons donc le bilan.

1.3.1 Effet optique

Si quelques précautions sont prises, les 4 afficheurs fourniront bien les 4 valeurs qui leur sont effectivement destinées et, de plus, paraîtront activés en permanence alors qu'il n'en est rien.

Pour celà il faut savoir jouer avec le phénomène de persistance rétinienne comme ont su le faire les inventeurs du cinéma : une image lumineuse persiste sur notre rétine pendant environ 20 millisecondes.

Il faudra donc réactiver chacun des afficheurs au moins 50 fois par seconde pour que le chiffre qu'il affiche ne nous paraisse pas clignoter. C'est du reste pour cette raison que le réseau d'éclairage est à une fréquence de 50 hertz.

Comme nous voulons présenter 4 chiffres quasi-simultanément, la fréquence de « balayage » des 4 afficheurs étant de 50 Hz, on aura 5 millisecondes à consacrer à chaque afficheur. Ce temps correspondra à la période du multivibrateur commandant l'horaire de multiplexage et de démultiplexage. Ce multivibrateur doit donc avoir une fréquence minimum de 200 Hz.

1.3.2 Économies

Les économies se traduisent par la réduction du nombre des décodeurs de 4 à 1 et par la réduction du nombre de fils entre le départ et l'arrivée. Ici 8 fils suffiront si l'on compte les 4 fils de la voie commune, les 2 fils d'horaire et les fils de l'alimentation.

En contre-partie il faudra rajouter les 20 portes du multiplexeur, l'oscillateur et le compteur binaire des époques et les 4 portes du démultiplexage.

Le bilan global ne paraîtrait pas positif si les constructeurs de circuits intégrés ne venaient à notre aide.

1.4 Circuits intégrés pour le multiplexage étudié

1.4.1 Circuits à 4 entrées et 1 sortie

Pour résoudre le problème examiné ici, les constructeurs nous proposent, en circuit TTL, le

74153 doté de 2 multiplexeurs à 4 lignes d'entrée pour 1 sortie.

Les 2 commandes de sélection A et B sont communes aux 2 multiplexeurs. Elles correspondent aux bits de comptage (A pour le poids le plus faible et B pour le plus fort) qui déterminent les 4 époques d'envoi des données sur le fil de sortie (*cf. fig.* 5. 7).

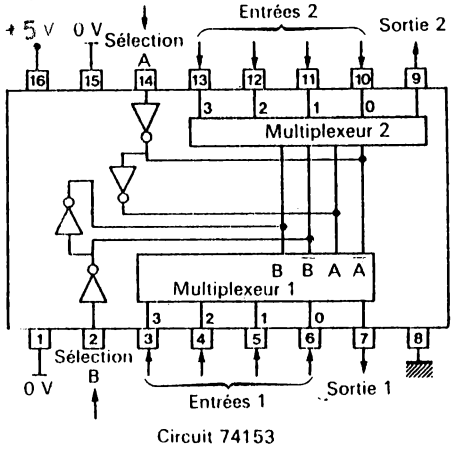


Fig. 5. 7

Deux circuits 153, proposant donc 4 multiplexeurs, vont permettre le multiplexage des 4 chiffres à afficher.

Reste un problème de câblage qui fait appel à beaucoup d'attention.

1.4.2 Réalisation du circuit de multiplexage

Dans les bascules de départ groupées par blocs de 4 (G_0, G_1, G_2 et G_3) nous avons les 16 valeurs binaires déjà figurées (*cf. fig.* 5. 8).

A la première époque (E_0), il faut faire parvenir sur les 4 sorties les valeurs A_0, B_0, C_0 et D_0.

Comme chaque multiplexeur ne peut envoyer qu'une seule valeur, A_0 doit figurer sur la sortie du premier multiplexeur (M_0), B_0 sur la sortie de M_1, C_0 sur M_2 et D_0 sur M_3.

Chacune de ces valeurs doit donc être amenée sur la première entrée de chaque multiplexeur.

En poursuivant le raisonnement on voit que les valeurs A_1, B_1, C_1 et D_1 doivent être amenées sur la deuxième entrée de chacun des multiplexeurs, A_2, B_2, C_2, D_2 sur la troisième et enfin A_3, B_3, C_3, D_3 sur la quatrième.

Le plan de câblage qui en résulte figure sur la figure 5. 8. Si on veut le réaliser en circuit imprimé il est indispensable d'utiliser la technique du « double face ».

1.5 Circuits intégrés pour le démultiplexage étudié

Le problème est relativement facile à résoudre par quelques portes comme nous l'avons vu. Toutefois il est encore plus simple de faire appel à un circuit tout prêt.

Rappelons qu'il s'agit, à partir des 2 bits du compteur de temps, de disposer de 4 sorties à niveau haut passant au niveau bas pendant l'époque où l'afficheur correspondant doit être activé.

L'étude générale de tels circuits a déjà été faite au chapitre 2 (*cf.* § 5.4 et 5.7) et nous y avons présenté le circuit 74154 comme un décodeur-démultiplexeur.

Ici ce sera un circuit analogue, appelé décodeur BCD-décimal, qui répondra au problème. Nous choisirons le 74145 par exemple (*cf. fig.* 5. 9 *a*) en

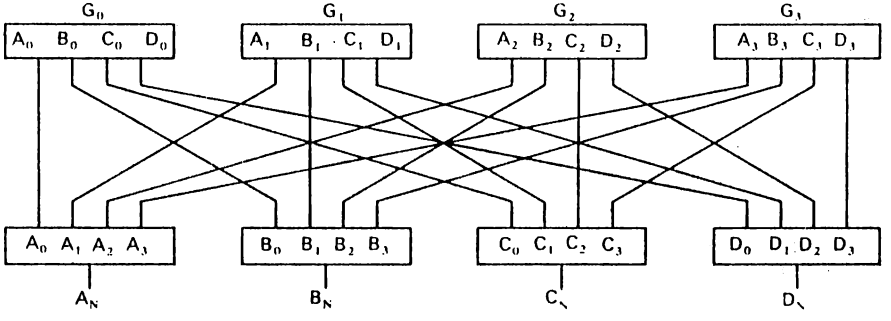


Fig. 5. 8

mettant les entrées C et D à la masse pour ne nous servir que des entrées A et B venant du compteur.

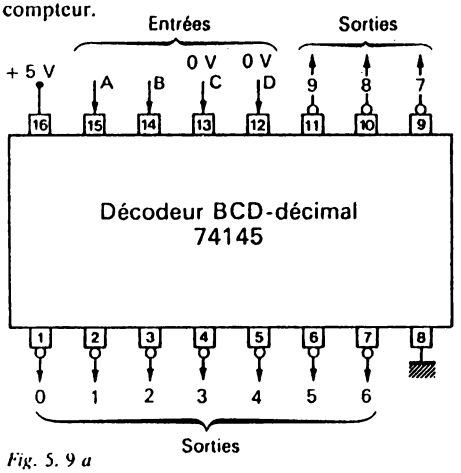


Fig. 5. 9 a

En nous référant à la table de vérité (*fig. 5. 9 b*) nous voyons que nous obtenons bien le créneau négatif recherché sur chaque sortie 0 à 3.

D	C	B	A	0	1	2	3	...
0	0	0	0	L	H	H	H	...
0	0	0	1	H	L	H	H	...
0	0	1	0	H	H	L	H	...
0	0	1	1	H	H	L	L	...

Fig. 5. 9 b

De tels circuits portent des noms variés tels que :

- décodeurs BCD-décimal
- décodeurs binaire-décimal
- démultiplexeurs
- sélecteurs 1 sortie parmi 10, parmi 16, etc.
- sélecteurs d'adresse.

C'est-à-dire qu'ils sont d'usage très fréquent en électronique digitale et que nous aurons d'autres occasions de les rencontrer.

1.6 Circuits généraux de multiplexage

Avec 2 entrées de compteurs on pourra faire sortir sur un seul fil, en série, 4 valeurs binaires (2^2).

Avec 3 entrées on peut compter jusqu'à 8, donc envoyer 8 valeurs en série sur la sortie.

Donc, de façon générale un multiplexeur enregistre 2^n valeurs binaires qu'il envoie sur la sortie, en série, au rythme d'un compteur doté de n fils de commande. Nous devons donc trouver des circuits intégrés à 2 entrées-1 sortie avec 1 fil de sélection, à 4 entrées-1 sortie-2 sélections, à 8 entrées-1 sortie et 3 sélections, à 16 entrées-1 sortie et 4 sélections.

Au-delà il sera difficile de présenter des circuits, le nombre de leurs « pattes » étant trop élevé.

C'est effectivement ce que nous trouvons dans les catalogues TTL.

Multiplexeurs 2 entrées-1 sortie :

le 74157 (quadruple)
le 74158 (quadruple) à sortie inversée
le 74257 (quadruple)
le 74258 (quadruple) à sortie inversée
le 74298 (quadruple)
le 74399 (quadruple)

Notons que la configuration des circuits 298 et 399 est particulièrement adaptée à la sélection successive, sur leur 4 sorties, de l'un ou de l'autre des 2 groupes de 4 informations d'entrée ($A_1B_1C_1D_1$ et $A_2B_2C_2D_2$).

Multiplexeurs 4 entrées-1 sortie :

le 74153 (double)
le 74253 (double)
le 74352 (double) sorties inversées
le 74353 (double) sorties inversées

Multiplexeurs 8 entrées-1 sortie :

le 74151 sorties directes et inversées
le 74152 sorties inversées
le 74251 sorties directes et inversées
le 74351 (double avec 4 entrées communes)

Multiplexeurs 16 entrées-1 sortie :

le 74150 sorties inversées

1.7 Circuits généraux de démultiplexage

Ici, pour une entrée comportant n fils de compteurs permettant de compter en binaire de 0 à 2^n, nous aurons 2^n sorties. Toutefois certains démultiplexeurs adaptés au code BCD auront 10 sorties pour 4 entrées.

Les circuits TTL proposés sont les suivants :

- *démultiplexeur* 2 entrées-4 sorties

le 74139 (double)
le 74155 (double)
le 74156 (double) à collecteur ouvert

les 155 et 156 sont facilement transformables en démultiplexeurs 3 entrées 8 sorties

– *démultiplexeur* 3 entrées-8 sorties
le 74138

– *démultiplexeur* 4 entrées-10 sorties (BCD → décimal)
le 74145

– *démultiplexeur* 4 entrées-16 sorties (Binaire → Décimal)
le 74154
le 74159 à collecteur ouvert.

1.8 En résumé

Multiplexer des données consiste, ces données se présentant en parallèle à un instant donné, à dissocier dans le temps l'envoi de ces données sur un seul conducteur. Le multiplexage vise à économiser les voies de transmission.

Démultiplexer les données multiplexées consiste à trier ces données en fonction du temps pour les affecter aux récepteurs qui leurs sont destinées.

Remarque :

Il n'est pas obligatoire que le temps intervienne dans le multiplexage. En électronique analogique on peut multiplexer plusieurs communications téléphoniques en utilisant des fréquences porteuses différentes, modulées en amplitude par ces communications, l'ensemble des fréquences modulées étant mélangées sur un même fil. Le démultiplexage consistera à filtrer ensuite chaque fréquence porteuse. Dans ce même esprit on peut dire que l'onde radio-électrique que capte l'antenne d'un poste radio est une voie de multiplexage et le dispositif d'accord du poste est le démultiplexeur (ou un élément de démultiplexage).

2. REGISTRES À DÉCALAGE

Le multiplexeur utilisé dans l'exemple précédent enregistre quatre données en parallèle et les restitue, l'une après l'autre, c'est-à-dire *en série*, grâce à des portes qui sont activées tour à tour par les signaux d'un compteur.

Si ces données sortent par un seul fil, c'est grâce à l'adjonction d'un circuit OU qui relie les 4 sorties du multiplexeur à ce conducteur unique.

Pour obtenir le même résultat, on peut faire appel à une technique différente : celle des registres à décalage (en anglais *SHIFT-REGISTERS*).

Le principe des registres à décalage peut être illustré par une allégorie.

Imaginons une boîte allongée divisée en n casiers (4 sur la figure 5. 10). Dans chaque casier est placé un scarabée.

Grâce à un dispositif on peut alors faire basculer les parois séparant les cases, ce qui permet, peu après, de voir les scarabées sortir de la boîte à la queue leu leu.

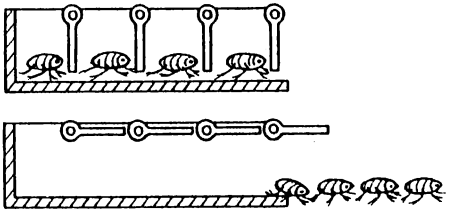


Fig. 5. 10

C'est ce chargement en parallèle et cette sortie en série qui caractérisent les registres à décalage. Quant au dispositif de décloisonnement on imagine facilement qu'il concerne les liaisons entre la sortie d'une cellule de registre et l'entrée dans la suivante.

2.1 Chargement en parallèle

En se référant à l'analogie précédente on peut distinguer deux phases essentielles dans le fonctionnement de ces registres : le chargement en parallèle des informations et leur sortie en série.

Nous utiliserons des bascules RS et imaginerons des registres à 4 bascules.

Les données à enregistrer, de valeurs a, b, c et d, seront amenées sur les entrées S (Set), l'entrée Reset étant reliée à S par un inverseur (*fig.* 5. 11).

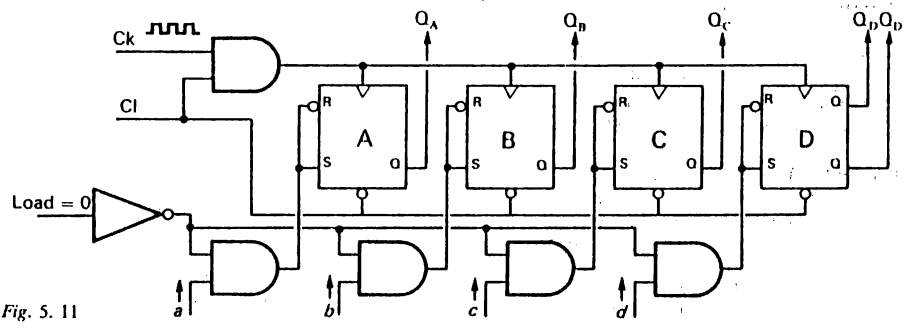


Fig. 5. 11

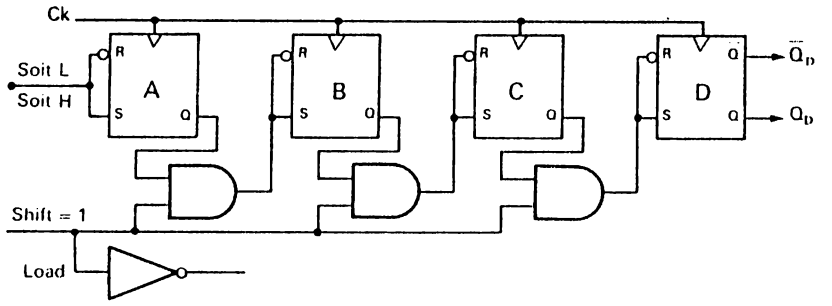


Fig. 5. 12

Dans ces conditions, à l'issu d'un front montant appliqué sur l'entrée «horloge» commune, la sortie de chaque bascule (Q_A à Q_D) délivrera la valeur appliquée sur S.

Ce coup d'horloge permet de rendre effectif le chargement en parallèle des données. Les scarabées sont dans les boîtes!

2.2 Le décalage

Nous allons relier la sortie Q de chaque bascule à l'entrée S de la suivante.

Cette fois, au coup d'horloge, la bascule B se chargera de la valeur figurant sur Q_A, la bascule C de la valeur de Q_B, etc. à condition qu'il n'y ait pas de conflit entre les entrées en parallèle et les liaisons entre bascules. Au moment où on relève les cloisons des boîtes (bascules), les accès du chargement en parallèle doivent être inhibés (*fig.* 5. 12).

Dans ce cas, les valeurs introduites dans les bascules vont, à chaque coup d'horloge, être décalées d'une bascule à droite. Sur la sortie Q_D on pourra recueillir successivement *d* (avant le premier coup d'horloge) puis *c*, *b* et enfin *a*. Ensuite, suivant que S_A aura été mis au niveau haut ou au niveau bas, on verra apparaître une série de 0 ou une série de 1, à moins qu'un autre chargement ne se produise.

2.3 Assemblage des deux fonctions

Les deux fonctions de chargement et de décalage doivent être regroupées dans un même circuit.

Chaque entrée S doit être en mesure de recevoir le bit des données d'entrée en parallèle si l'on est dans le mode «chargement». Sinon, elle doit être reliée à la sortie du registre qui la précède.

Il faut donc une commande de chargement, désignée généralement par la lettre L (LOAD), qui par sa valeur 0 ou 1 autorisera le chargement en inhibant le décalage ou autorisera le décalage en inhibant le chargement.

Dans ce dernier cas la commande prendra plutôt le nom de décalage et sera baptisée SHIFT.

Dans beaucoup de registres on retrouvera cette commande unique baptisée Shift/Load. Cette expression signifie qu'à l'état 1 la commande permet le décalage, et à l'état 0 elle permet le chargement (il vaudrait mieux écrire Shift/$\overline{\text{Load}}$).

Si L = 0, on doit avoir $S_A = \overline{L}a$, sinon $S_A = 0$ ou $S_A = 1$ suivant le câblage que l'on aura fait.

Pour la bascule B :

$$S_B = \overline{L}b + Q_A L$$

Pour la bascule C :

$$S_C = \overline{L}c + Q_B L$$

et enfin

$$S_d = \overline{L}d + Q_C L$$

Ces relations logiques sont réalisées par des portes (*fig.* 5. 13).

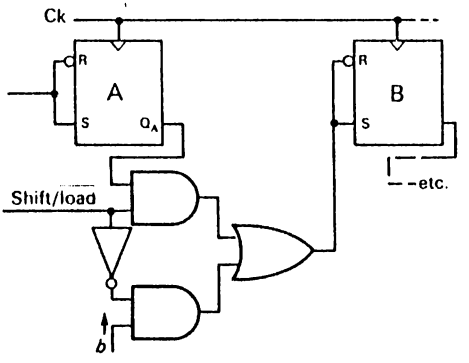


Fig. 5. 13

Nous obtenons le schéma de la figure 5.14 qui superpose les 2 fonctionnements et introduit quelques variantes, dont 1 entrée JK sur la première bascule (nous en étudierons l'utilité un peu plus loin).

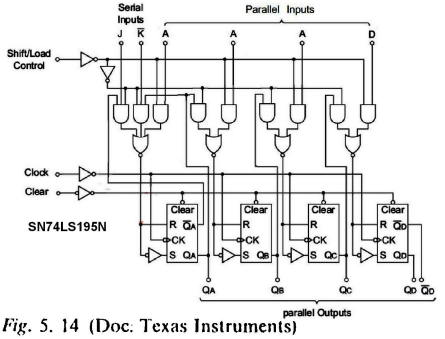


Fig. 5. 14 (Doc. Texas Instruments)

Remarque :

Le principe des registres à décalage a déjà été vu dans les exercices du chapitre 3 (réalisation d'un chenillard).

Ces registres peuvent être réalisés avec des bascules JK maître-esclave, des bascules D ou des bascules RS.

2.4 Registres à décalage en circuit intégré

Il existe, en technologie TTL, de nombreux circuits intégrés répondant au problème posé ici et disposant souvent de circuits complémentaires leur donnant des performances plus larges.

Ce sont les circuits 7495, 99, 165, 166, 178, 179, 194, 195, 198, 199, 295B, 299 et 323.

Parmi toutes les fonctions réalisables par l'un ou l'autre de ces circuits on peut noter :

- Entrée en parallèle, sortie en série.
- Entrée en série, sortie en parallèle.
- Entrée en parallèle, sortie en parallèle.
- Entrée en série, sortie en série.
- Décalage à droite (*abcd* donne *d*, puis *c*, puis *b*, puis *a*).
- Décalage à gauche (*abcd* donne *a*, puis *b*, puis *c*, puis *d*).
- Possibilité de mise en cascade des circuits.

Il est nécessaire d'étudier très en détail la fiche technique de chaque circuit si l'on veut l'utiliser correctement.

2.5 Le circuit 74195

Ce circuit est un des plus classiques. Nous proposons ici l'analyse de ses possibilités à titre d'illustration.

On trouvera son schéma de réalisation à la figure 5. 14.

Il se compose de 4 bascules RS commandées par un double circuit suivant que la borne Shift/Load est au niveau bas (chargement) ou au niveau haut (décalage).

2.5.1 Chargement en parallèle du 74195

Rappelons qu'en appliquant un niveau bas sur la borne Shift/Load, le circuit est ramené à celui de la figure 5. 11. Il faut noter que la borne « Clear » doit être impérativement mise au niveau haut si l'on veut que ce chargement ait lieu.

Dans ces conditions nous avons affaire à 4 bascules indépendantes, soumises à 1 horloge commune.

Dès le premier coup d'horloge suivant le passage au niveau bas de Shift/Load, les données sont enregistrées dans les bascules et apparaissent sur les sorties Q.

2.5.2 Décalage à droite

Lorsque le signal Shift/Load passe au niveau haut, le schéma équivalent du 195 devient celui de la figure 5. 12.

Chaque bascule B, C et D voit son entrée « Set » reliée à la sortie de la bascule précédente.

L'entrée de la première bascule dépend, elle (voir *fig.* 5. 14), de la valeur précédente de sa sortie Q_A (Q_{A_0}) et des valeurs appliquées en J et K :

$$R_A = J \cdot \overline{Q}_{A_0} + \overline{K} \cdot Q_{A_0}$$

Ce qui donne la table de vérité suivante :

Shift	J	$\overline{K}$	Ck	Q_A
H	L	L	↑	L
H	L	H	↑	$\overline{Q}_{A_0}$
H	H	L	↑	Q_{A_0}
H	H	H	↑	H

En ce qui concerne les bascules suivantes, à chaque front montant de l'horloge elles vont être chargées des valeurs présentes sur leur entrée, c'est-à-dire présentes sur la sortie de la bascule précédente. Ces valeurs vont être décalées chaque fois d'une position sur la droite.

2.6 Mise en œuvre du circuit 74195

Appliquons-nous à obtenir les mêmes résultats que ceux obtenus avec un multiplexeur : nous chercherons tout d'abord à charger en parallèle des valeurs binaires et à les envoyer en série sur un seul fil.

2.6.1 Chargement en parallèle

A priori cette opération ne présente pas de difficulté; il suffit de porter «Shift/Load» au niveau bas en vérifiant que Clear reste bien à 1 pour que, dès le premier coup d'horloge les bascules soient chargées.

Le problème tient au fait qu'en matière d'électronique digitale on recherche la plus grande rapidité possible pour une économie maximum de moyens grâce à une bonne organisation des opérations de traitement.

Si un seul créneau d'horloge suffit pour charger les registres il ne faut pas faire durer le chargement plus longtemps; on réservera les créneaux suivants au décalage. On pourra ainsi faire apparaître plus souvent des données nouvelles *abcd* que l'on pourra charger puis décaler pour les faire sortir en série.

Toutes ces opérations doivent être, en même temps, organisées pour que leur déroulement ait lieu de façon automatique.

Il faut donc que des conditions bien précises permettent de faire apparaître le créneau de chargement suivi des créneaux de décalage.

Une solution serait de disposer d'une seconde horloge donnant un créneau négatif au moment souhaité pour le chargement. Ce créneau pourrait être dimensionné grâce à un monostable (74121 par exemple); on répugne souvent à son emploi car il faut y ajouter un couple RC dont l'encombrement n'est pas négligeable et dont les caractéristiques risquent de varier au vieillissement.

Nous allons voir qu'il existe une solution plus élégante.

2.6.2 Décalage commandant le chargement

Analysons le fonctionnement du circuit présenté à la figure 5. 15.

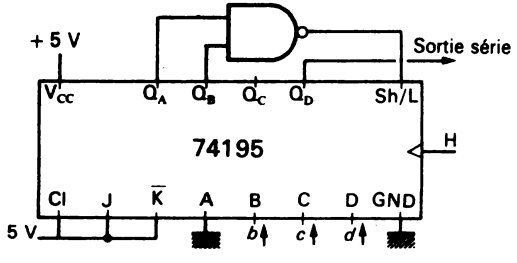


Fig. 5. 15

Clear, J et $\overline{K}$ sont au niveau 1.

Si $Q_A = 1$ et $Q_B = 1$, la porte NAND transmet un niveau bas sur Shift/Load, permettant le chargement des valeurs *abcd*.

Or *a* est égal à 0 (le conducteur est mis à la masse). Dès que le chargement est fait il apparaît donc un 0 sur Q_A et la porte NAND commute à 1 mettant le registre en position décalage.

Au front montant d'horloge suivant :

$Q_B = 0$ (décalage de Q_A)

$Q_A = 1$ car J et $\overline{K}$ sont au niveau haut

Au second front montant

$Q_C = 0$
$Q_B = 1$
$Q_A = 1$

Ces deux dernières valeurs provoquent un 0 sur Shift/Load et un nouveau chargement.

On peut dresser le tableau des états successifs à partir du premier chargement :

Horloge	Q_A	Q_B	Q_C	Q_D
$H_0\uparrow$	0	*b*	*c*	[*d*]
$H_1\uparrow$	1	0	*b*	[*c*]
$H_2\uparrow$	1	1	0	[*b*]
$H_3\uparrow$	0	*b*	*c*	*d*

Ainsi, en 3 coups d'horloge, nous avons obtenu le chargement en parallèle de 3 données binaires *bcd* (au temps H_0) et leur apparition successive en série sur la sortie Q_D (aux temps H_0 H_1 et H_2).

Mais pour cela nous avons dû sacrifier 1 bit de donnée, la valeur *a*, pour la remplacer par ce que l'on appelle un 0 *marqueur*. C'est en effet ce 0 qui empêche le passage au niveau bas de Shift/Load pendant les temps H_1 et H_2.

Cette solution ampute donc le circuit 195 d'un quart de sa capacité.

Nous allons voir comment y remédier.

2.6.3 Augmentation de la capacité des registres à décalage 74195

On peut mettre en cascade deux circuits 195. Il suffit pour cela de relier la sortie Q_4 du premier aux entrées J et $\overline{K}$ du second (*cf.* table de vérité du § 2.5.2). Le schéma de la figure 5. 16 donne un exemple de cette mise en cascade. La borne d'entrée A sert toujours au chargement du zéro marqueur. Il est alors possible de charger 7 bits (de *b* à *h*).

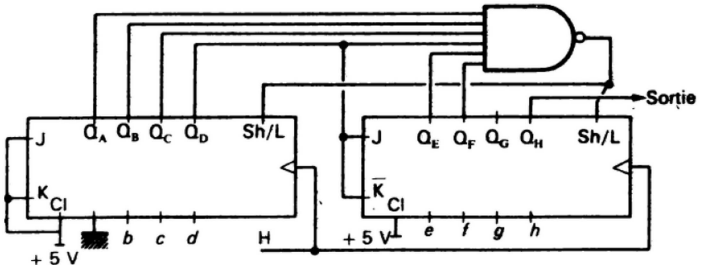


Fig. 5. 16

Pour que le chargement ait lieu, il faut que le 0 marqueur enregistré dans la première bascule ait parcouru toutes les bascules de A jusqu'à F et se trouve dans l'avant-dernière. A ce moment-là, la dernière donnée de sortie en série, *b* est déjà disponible sur la fil de sortie Q_H.

Pour avoir un registre de 8 bits, on peut par exemple rajouter une bascule en tête du dispositif précédent. On obtiendra alors le schéma de la figure 5. 17.

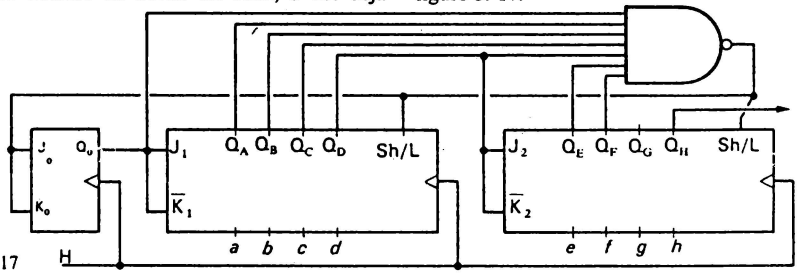


Fig. 5. 17

Le chronogramme (*fig.* 5. 18) nous précise le fonctionnement d'un tel montage.

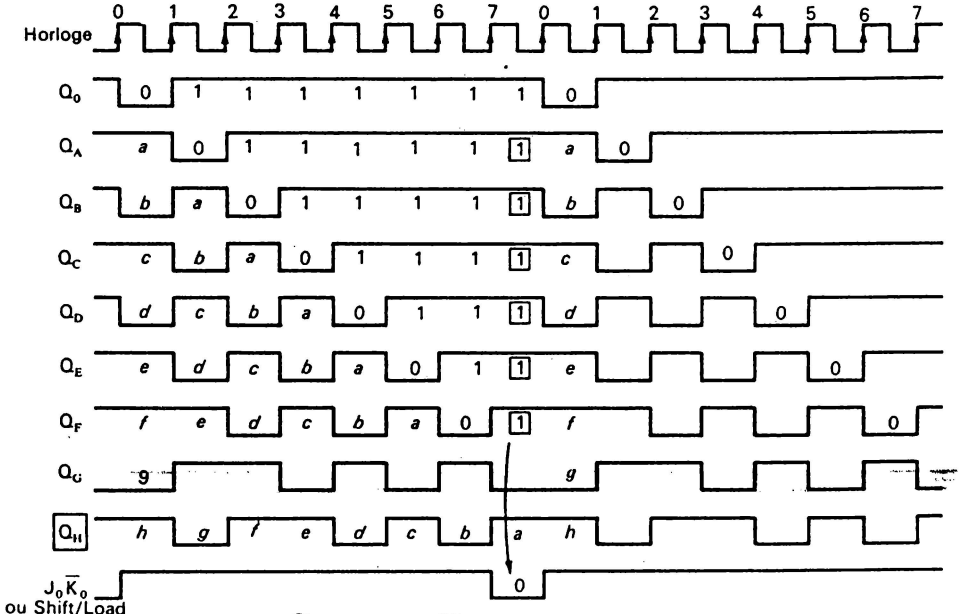


Fig. 5. 18

Chargement parallèle et sortie série sur Q_H du montage de la figure 5.17

2.6.4 *Généralisation du montage à un nombre quelconque de bits*

Grâce à ce type de montage nous pourrons charger en parallèle et sortir en série un nombre quelconque de bits, à conditions de respecter les capacités de sortance des portes (une sortie de porte LS peut alimenter 10 entrées).

Par exemple, 1 registre de 5 bits peut utiliser 2 registres 195. L'entrée A_0 servira pour le 0 marqueur, les entrées B_0 C_0 D_0 .A_1 et B, pour l'entrée des données.

La porte NAND aura 4 entrées (5 – 1), branchées sur Q_{A_0}, Q_{B_0}, Q_{C_0} et Q_{D_0} et la sortie en série se fera sur Q_{B_1}.

Ceci constitue un avantage non négligeable, dans certaines applications, sur les multiplexeurs qui sont liés aux puissances successives de 2 du système binaire (par 2, par 4, par 8, par 16).

2.7 Entrée « série » et sortie « série »

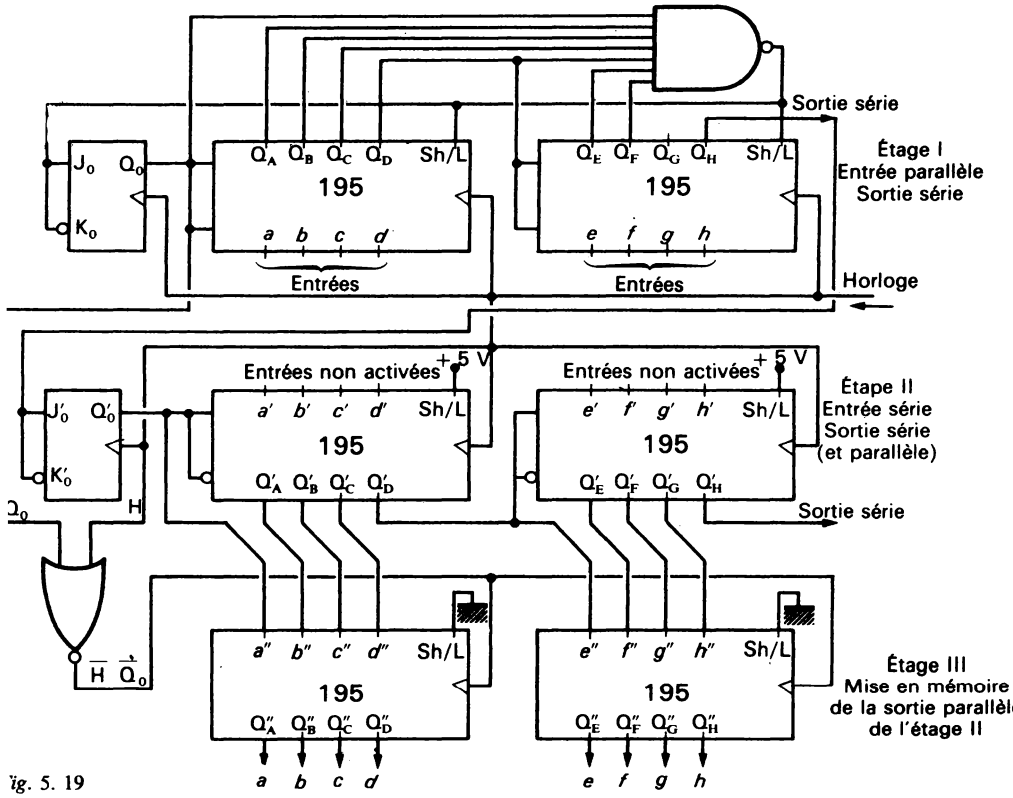
Le chargement en série du 195 se fait par l'entrée J(K), la commande « Shift/Load » étant à l'état haut.

Dans ce cas, à chaque coup d'horloge un décalage à droite se produit. Si cette horloge est en synchronisme avec l'horloge de l'exemple précédent, chaque valeur figurant en sortie du premier montage sera introduite dans la première bascule puis traversera tous les registres jusqu'au dernier. En sortie sur ce dernier registre on aura la sortie en série des valeurs.

A un moment et à un seul (correspondant à une période de l'horloge) toutes les valeurs de *a* à *h* (dans le cas de 8 bits) apparaîtront aux sorties Q_A à Q_H. Il n'en faut pas plus pour pouvoir saisir ces informations... à condition de bien déterminer ce moment.

2.8 Entrée « série » et sortie « parallèle »

Si l'on examine le chronogramme de la figure 5. 18, on y voit que lorsque la sortie Q_0 de la première bascule (bascule additive) passe à 0 (deuxième ligne du chronogramme), toutes les valeurs chargées, de *h* à *a*, ont été transmises à l'entrée de l'étage 2 (à entrée série), et la dernière de ces valeurs, *a*, figure déjà sur la sortie Q'_0 de la bascule additive de ce deuxième étage (*cf. fig.* 5. 19).



ig. 5. 19

En effet, la première valeur *h* d'un nouveau train de 8 bits est sur la sortie Q_H du premier étage, à la suite d'un nouveau chargement. Cette même valeur est donc présente sur l'entrée J'_0 du deuxième étage.

Relions toutes les sorties Q des circuits 195 du second étage (de Q'_0 à Q'_G) aux entrées de données d'un nouvel étage de circuits 195 (sans bascule additionnelle) dont les commandes Shift/Load sont en permanence au niveau bas. Les 8 bascules de ces registres sont indépendantes et vont servir à retenir (mettre en mémoire) les valeurs apparaissant sur les sorties Q du deuxième étage, au moment approprié.

Ce moment correspond au milieu de la période pendant laquelle Q_0 (étage I) est au niveau bas. En associant Q_0 et l'horloge sous la forme $\overline{Q_0}\,\overline{H}$, on obtiendra un front montant qui sera situé précisément à ce moment-là et qui activera le chargement des 2 registres de l'étage III.

Remarque :

Ce troisième étage n'est pas indispensable et ne sert qu'à maintenir plus longtemps les valeurs *a* à *h*.

On peut aussi appliquer le signal $\overline{Q_0}$ sur des portes AND d'autorisation placées sur les sorties Q'_0 à Q'_G du deuxième étage pour obtenir ces mêmes valeurs, mais de façon plus fugace.

2.9 En conclusion

L'exemple traité ici montre la polyvalence des circuits dits « registres à décalage ». Il permet aussi de juger de l'opportunité de multiplexer ou de mettre en série par registres à décalage.

Dans le cas présent, si l'on ne tient pas compte de l'alimentation, on voit (*fig.* 5. 19) qu'il suffit de 3 fils pour relier l'étage I aux étages II et III qui pourront, sans trop de problèmes, se situer à distance.

S'agissant de multiplexer 8 valeurs binaires, il faudrait le fil de multiplexage et 3 fils de comptage. La différence n'est pas notoire.

L'avantage essentiel du décalage résidera surtout, dans la possibilité de traiter une quantité quelconque de valeurs binaires (3, 5, 6, 7, 9, 10 etc.). Et puis l'on découvrira, dans les exercices de ce chapitre, d'autres possibilités de ces registres à décalage.

3. DÉCALAGE À GAUCHE

Dans certaines applications que nous aborderons plus loin, il est nécessaire de procéder à des décalages à gauche.

3.1 Décalage à gauche avec un registre à décalage à droite

Si l'on doit affecter à cette opération un registre spécialisé, on peut fort bien utiliser un registre à décalage à droite.

En effet, soit le nombre *abcd*. Décalé à droite il donnera 0, *a*, *b*, *c*, en supposant que les places vides sont occupées par des 0. Décalé à gauche nous aurions dû obtenir le nombre *bcd*0.

Il suffit, pour obtenir ce résultat, d'introduire dans le registre à décalage à droite le nombre *dcba*. Décalé 1 fois, il donnera 0*dcb* qu'il suffit de lire *à l'envers*.

Pour décaler à gauche il suffira donc d'introduire, à l'envers, le nombre à décaler, puis de lire à l'envers encore ce même résultat.

3.2 Décalage à droite ou à gauche

Un même registre peut jouer alternativement le rôle de registre à décalage à droite *ou* à gauche (il s'agit cependant d'un ou exclusif).

On peut obtenir ce résultat avec le 74195, en reliant les sorties Q_B Q_C Q_D respectivement aux entrées *a*, *b* et *c*.

Lorsque le signal Shift/Load sera au niveau bas, nous aurons affaire à un registre à décalage à gauche, la sortie des informations se faisant en série sur Q_A.

Avec un signal Shift/Load à 1, le registre continuera à décaler à droite, puisque dans ce cas les entrée a, b, c, d sont déconnectées. La sortie se fera, en série, sur Q_D.

3.3 Circuit universel 74198

Si l'on veut tout à la fois charger en parallèle ou en série, lire en parallèle ou en série, décaler à droite ou à gauche, on pourra faire appel au circuit 74198, où tous les câblages et toutes les portes (87 au total) sont intégrés.

La figure 5. 20 donne les caractéristiques de ce circuit à 24 pattes.

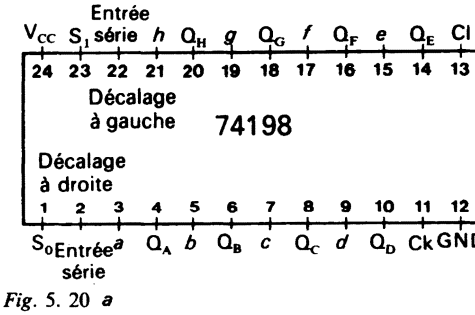


Fig. 5. 20 *a*

Table de fonctions du 74198

Entrées							Sorties
Clear	S_1	S_0	Ck	Entrée série		Entrée parallèle	Q_A Q_B ... Q_G Q_H
				Gauche	Droite	*a* à *h*	
L	X	X	X	X	X	X	L L ... L L
H	X	X	L	X	X	X	Q_{A_0} Q_{B_0} ... Q_{G_0} Q_{H_0}
H	H	H	↑	X	X	*a* ... *h*	*a* *b* ... *g* *h*
H	L	H	↑	X	H	X	H Q_{AN} ... Q_{FN} Q_{GN}
H	L	H	↑	X	L	X	L Q_{AN} ... Q_{FH} Q_{GN}
H	H	L	↑	H	X	X	Q_{BN} Q_{CN} ... Q_{HN} H
H	H	L	↑	L	X	X	Q_{BN} Q_{CN} ... Q_{HN} L
H	L	L	X	X	X	X	Q_{A_0} Q_{B_0} ... Q_{G_0} Q_{H_0}

Fig. 5. 20. b

EXERCICES

1. *Exercice guidé*

Utilisation du registre à décalage 74195 comme diviseur par N.

Comme nous l'avons vu lors des exercices du chapitre 3, un registre à décalage peut être utilisé comme diviseur par un nombre quelconque.

L'application proposée ici sera un peu différente et permettra de faire apparaître quelques caractéristiques nouvelles de ces registres à décalage.

Il s'agit d'utiliser les entrées J et $\overline{K}$ du circuit 195 pour modifier, lors des décalages successifs, les valeurs qui vont s'inscrire dans la bascule de gauche.

a) Rappel des conditions d'entrée sur J et $\overline{K}$

Celles-ci figurent dans le tableau du § 2,52 de ce chapitre. Nous les rappelons ici :

Shift	J	$\overline{K}$	Ck	Q_A
H	L	L	↑	L
H	L	H	↑	$\overline{Q_{A_0}}$
H	H	L	↑	$\overline{Q_{A_0}}$
H	H	H	↑	H

b) Bouclage de la valeur de droite du registre sur J et $\overline{K}$

Soit $Q_A Q_B ... Q_N$ les valeurs apparaissant après un coup d'horloge sur le registre 195, la commande Shift étant à 1.

Si nous relions Q_N à $\overline{K}$ et $\overline{Q}_N$ à J, lorsque $Q_N = 0$, nous aurons $J = 1$ et $\overline{K} = 0$ la nouvelle valeur Q_{A_1}, au prochain coup d'horloge sera égale à $\overline{Q_{A_0}}$.

Lorsque $Q_N = 1$, nous aurons $J = 0$ et $\overline{K} = 1$ la prochaine valeur Q_{A_1} sera égale à Q_{A_0}.

Par exemple, si le nombre contenu dans le registre est 0 1 0 0, au coup suivant il deviendra 1 0 1 0. En effet :

Q_A Q_B Q_C Q_D	Commentaires
0 1 0 0 1 0 1 0	Le zéro de droite va transformer $Q_A = 0$ en $Q_A = 1$, Les autres valeurs sont décalées vers la droite.

c) Séquences obtenues avec cinq registres

Nous allons utiliser deux circuits 195. Nous n'utiliserons que la première bascule du second circuit pour ne pas rallonger les démonstrations, mais l'exemple est généralisable à 6, 7 ou 8 bascules.

Ces deux circuits sont reliés conformément au montage de la figure E. 5. 1.

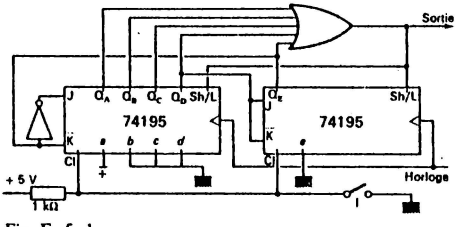


Fig. E. 5. 1

A la mise sous tension nous mettons les bascules à 0 en agissant sur l'interrupteur I, puis en le relâchant.

L'ensemble des 0 sur les sorties met la commande Shift/Load à 0 à travers la porte OR à 5 entrées, ce qui provoque le chargement de la valeur 10000.

Cette nouvelle valeur porte Shift/Load à 1 et les décalages commencent suivant le cycle suivant (application de ce que nous venons d'étudier en *b*).

1 0 0 0 0	0 0 1 1 0	1 1 1 0 1
0 1 0 0 0	1 0 0 1 1	1 1 1 1 0
1 0 1 0 0	1 1 0 0 1	0 1 1 1 1
0 1 0 1 0	1 1 1 0 0	0 0 1 1 1
1 0 1 0 1	0 1 1 1 0	0 0 0 1 1
1 1 0 1 0	1 0 1 1 1	0 0 0 0 1
0 1 1 0 1	1 1 0 1 1	0 0 0 0 0

Cette dernière valeur remet Shift/Load à 0 et le cycle des 21 valeurs recommence.

Nous avons en effet obtenu 21 configurations successives et différentes des cinq valeurs binaires Q_A, Q_B, Q_C, Q_D et Q_E.

Peu importe la traduction de ces valeurs en décimal, ce à quoi nous allons tout d'abord nous intéresser est le signal obtenu sur la sortie de la porte OR (signal de commande du Shift/Load).

Ce signal passe à 0 tous les 21 coups d'horloge. Il constitue donc un diviseur par 21 du signal d'horloge.

d) Division par un nombre compris entre 2 et 21

Si nous chargeons, à la place de la valeur 10000, une des valeurs rencontrées pendant ce cycle, nous obtiendrons un nouveau diviseur, par un nombre inférieur à 21.

Pour choisir à coup sûr le nombre diviseur, il suffit de numéroter chaque configuration de 21 à 1 comme ci-dessous :

N°	*Configuration*
21	1 0 0 0 0
20	0 1 0 0 0
19	1 0 1 0 0
18	0 1 0 1 0
17	1 0 1 0 1
16	1 1 0 1 0
15	0 1 1 0 1
14	0 0 1 1 0
13	1 0 0 1 1
12	1 1 0 0 1
11	1 1 1 0 0
10	0 1 1 1 0
9	1 0 1 1 1
8	1 1 0 1 1
7	1 1 1 0 1
6	1 1 1 1 0
5	0 1 1 1 1
4	0 0 1 1 1
3	0 0 0 1 1
2	0 0 0 0 1
1	0 0 0 0 0

Il suffit alors de se référer au numéro affecté, il indique le diviseur obtenu.

En modifiant uniquement les entrées *a b c d e* nous obtenons par exemple :

pour $a = 0$ $b = 0$ $c = 1$ $d = 1$ $e = 0$
un diviseur par 14

pour $a = 1$ $b = 0$ $c = 1$ $d = 1$ $e = 1$
un diviseur par 9.

Remarque : Toutes les configurations possibles de cinq valeurs binaires ne figurent pas dans cette séquence.

On peut en effet constater que l'on peut obtenir deux autres séquences de valeurs différentes en partant

d'une part de 00010 (7 valeurs différentes),

d'autre part de 00100 (3 valeurs différentes),

et enfin que le nombre 11111 s'auto-entretient, et qu'il ne présente pas ici d'intérêt puisqu'il bloque apparemment le système.

En ajoutant ces configurations (21 + 7 + 3 + 1) on obtient bien les 32 configurations possibles.

e) Diviseur programmable

Si l'on peut, par un procédé manuel ou automatique, afficher en *abcde* les valeurs binaires correspondant à l'un ou l'autre des numéros de 21 à 2, on aura obtenu ce que l'on appelle un « diviseur programmable ».

2. Imaginer un dispositif manuel, par contacteur à glissière, permettant de « programmer » le diviseur qui vient d'être décrit.

3. *a)* A partir d'un signal d'horloge H_0, on voudrait produire un nouveau signal H_1 représenté sur la figure E. 5. 2.

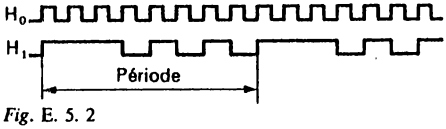


Fig. E. 5. 2

Comment peut-on s'y prendre?

Envisager différentes solutions : multiplexeur ou registre à décalage (choisir le 195).

b) Si l'on observe le tableau des 21 configurations numérotées (exercice 1, *d*), on constate que, dans les configurations 21 à 14, la sortie Q_D prend les valeurs 00010101 successivement.

Comment exploiter cette propriété pour répondre au problème posé en *a*?

4. Réaliser un décodeur-démultiplexeur 1 sortie parmi sept en utilisant deux registres 74195.

Les sorties devront être au niveau haut pendant six septièmes du temps et basses au moment où elles sont activées.

CHAPITRE 6

Les mémoires à accès aléatoire

1. MÉMOIRES ET RANGEMENT

Si l'on a souvent comparé l'ordinateur à un cerveau humain, c'est en grande partie parce qu'on a été amené à constater qu'on ne pouvait pas faire de machine à traiter les informations sans les doter de très nombreuses cellules de mémoire, un peu analogues aux neurones du cerveau.

On ne peut, en effet, faire effectuer une opération de calcul ou un simple transfert d'information à un circuit électronique s'il n'existe pas de registres qui stockent les nombres ou les informations à traiter ou bien les résultats.

De plus, ayant doté le système de ces mémoires, il est nécessaire encore de pouvoir retrouver l'information à traiter au moment voulu, parmi les milliers d'autres informations qui auront pu être enregistrées. C'est pourquoi, au cours de ce chapitre, nous aborderons les deux aspects inséparables de *mémoire* et d'*adresse* de mémoire.

1.1 L'élément de mémoire : la bascule bistable

Un élément de mémoire est la cellule unitaire permettant d'enregistrer une valeur binaire 0 ou 1. On pourra parler de cellule de mémoire, de position de mémoire ou de registre de mémoire, ce dernier terme étant très souvent employé.

Ce registre de mémoire a déjà été évoqué très largement dans les chapitres précédents sous forme de bascule bistable : bascules D, RS, T, ou maître-esclave.

Les circuits TTL de la série 74 proposent des assemblages de 2, 4 ou 8 bascules de ces types.

Mais pour traiter des informations il est nécessaire de disposer de milliers ou plutôt de dizaines et centaines de milliers de mémoires.

Depuis que les ordinateurs ont été inventés, le génie humain s'est attaché à intégrer de plus en plus de cellules de mémoires dans le minimum de place, pour un minimum de consommation en énergie.

Nous vous proposons de suivre, dans le paragraphe suivant, un historique en raccourci de cette recherche.

1.2 Historique sur la réalisation des mémoires

1.2.1 Mémoires à relais

Le simple interrupteur qui commande l'allumage ou l'extinction d'une ampoule électrique est une mémoire : il conserve l'information qu'il a reçue par la pression d'un doigt.

En électronique digitale il faudra remplacer la pression du doigt par l'apparition d'une tension sur un conducteur. L'idée qui vient alors est d'utiliser un électro-aimant pour agir sur l'interrupteur, c'est-à-dire de constituer un relais électro-magnétique.

Ce relais sera commandé directement par la tension (l'information) à mettre en mémoire (voir *fig.* 6. 1).

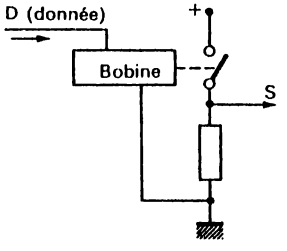


Fig. 6. 1

Mais ce circuit ne constitue pas une véritable mémoire : dès que la tension de commande disparaît, le relais se déclenche à nouveau, ouvrant le circuit.

Il faut donc imaginer un dispositif d'auto-entretien. En contrepartie il faudra pouvoir remettre à 0 le système pour une nouvelle mise en mémoire.

On obtient alors un circuit du type de celui de la figure 6. 2. Une commande positive sur D ferme le circuit passant par I_1 et I_2 et une tension apparaît sur S, maintenant le circuit fermé. On comprend aisément le rôle joué par la diode.

Si l'on veut remettre à 0 le circuit, il faudra agir sur le deuxième relais par la commande $\overline{D}$.

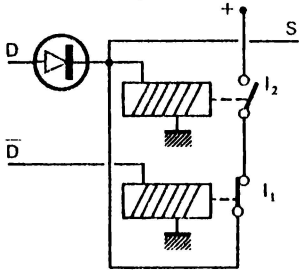


Fig. 6. 2

En disposant sur la sortie S un amplificateur non inverseur et un amplificateur inverseur, nous avons le schéma type d'une mémoire (*fig.* 6. 3) avec ses 2 entrées D et $\overline{D}$ (ou S et R) et ses 2 sorties S et $\overline{S}$ (ou Q et $\overline{Q}$).

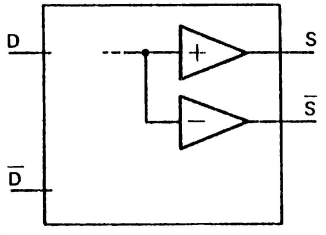


Fig. 6. 3

Ce registre à relais n'a, de nos jours, qu'un intérêt historique. Sa partie la moins fiable est constituée par les pièces mécaniques en mouvement. Malgré son encombrement, son coût et son manque de fiabilité il a été utilisé dans les premiers ordinateurs.

1.2.2 Mémoires magnétiques

Face aux pannes trop fréquentes provoquées par les pièces mécaniques, on peut imaginer que des cerveaux créatifs ont eu l'idée de supprimer tout simplement ces pièces.

Si l'on ne garde que la bobine du relais, rappelons que le noyau de cette bobine (généralement en fer doux) va s'aimanter dans une direction Nord-Sud ou Sud-Nord en fonction du sens du courant qui traverse la bobine.

Mais le phénomène le plus intéressant est le phénomène d'hystérésis.

Si l'on ne fait plus passer de courant dans la bobine, le noyau reste aimanté faiblement, mais il garde en mémoire le sens de sa dernière aimantation.

Lorsque la bobine est parcourue par un courant + I, le noyau acquiert une aimantation N-S (Nord-Sud), par exemple, figurée par le point A sur la figure 6. 4.

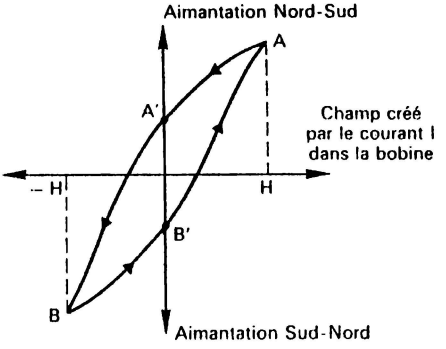


Fig. 6. 4

Si le courant de la bobine est interrompu, l'aimantation du noyau ne disparaît pas complètement. Elle diminue et est représentée par l'ordonnée du point A'.

Si l'on fait passer un courant inverse − I, l'aimantation s'inverse et atteint le point B, pour revenir au point B' si le courant, dans la bobine, devient nul.

Voici comment ce phénomène a été exploité dans la réalisation de mémoires :

Comme il n'y a plus à faire fonctionner de contacts, le noyau peut être refermé sur lui-même, en forme de tore, c'est-à-dire d'anneau. C'est une bonne façon de conserver plus longtemps l'aimantation.

Un bobinage b_1 recevant une impulsion, il restera après le passage de cette impulsion, une aimantation «rémanente» qui est la mémoire (*fig.* 6. 5).

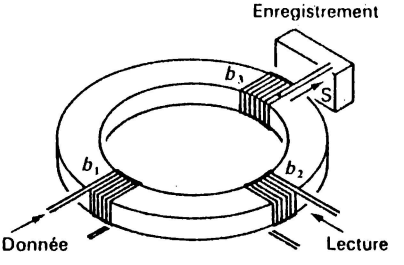


Fig. 6. 5

Un deuxième bobinage b_2, en sens inverse du premier servira à la lecture. En envoyant une impulsion sur ce bobinage, l'aimantation de l'anneau sera inversée. Elle prendra en fait successivement les valeurs représentées par le parcours A', B, B' de la figure 6. 4.

Pour concrétiser cette lecture il est nécessaire de disposer un nouveau bobinage b_3. Les variations d'aimantation du noyau induiront dans cette bobine de petites variations de courant dont l'allure est représentée sur la figure 6. 6 *a*.

Si la donnée qui a été mise en mémoire est un 0, la « lecture » amènera l'aimantation du point B' au point B puis à nouveau en B' ce qui donnera une réponse telle que celle de la figure 6. 6 *b*.

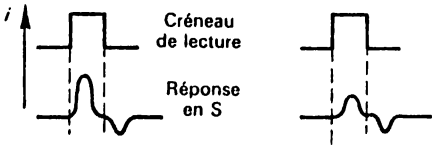


Fig. 6. 6 *a* *b*

La différence entre les deux lectures n'est pas facile à discerner. C'est pourquoi les recherches se sont orientées vers des matériaux ferromagnétiques dont la courbe d'hystérésis soit la plus carrée possible.

Des alliages spéciaux ont permis d'obtenir des courbes d'hystérésis telles que celle de la figure 6. 7. Les lectures, dès lors, sont très nettement différenciées (*fig.* 6. 8).

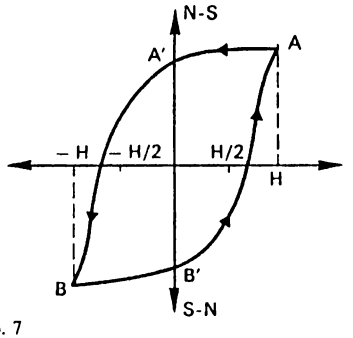


Fig. 6. 7

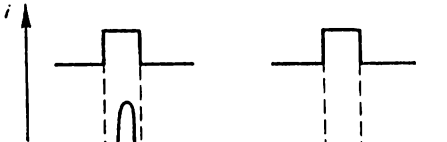


Fig. 6. 8

De tels tores, de 4 à 5 millimètres de diamètre, ont été utilisés. La confection des bobines s'apparentait aux travaux de broderie.

Aussi, pour simplifier le travail de tissage, et diminuer la taille des tores a-t-on été amené à se contenter d'un demi-tour de fil, c'est-à-dire à son passage pur et simple à travers le tore. Les faibles énergies mises en jeu étant largement amplifiées à la sortie (*fig.* 6. 9).

La solution la plus originale fut surtout le groupement de ces tores de ferrite en *matrices*.

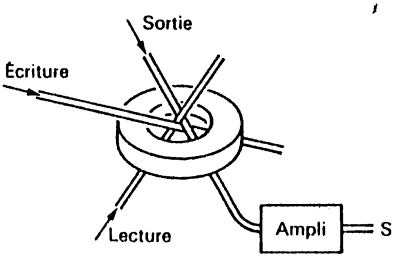


Fig. 6. 9

1.2.3 *Matrices de tores de ferrite*

Si l'on groupe 64 tores de ferrite en une matrice carrée de 8 lignes et 8 colonnes nous obtenons la disposition de la figure 6. 10.

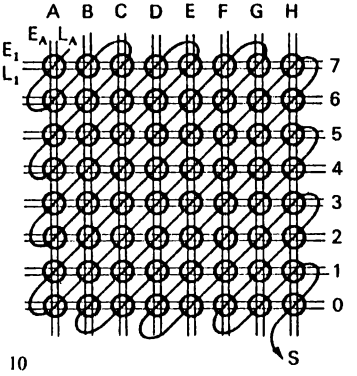


Fig. 6. 10

A travers chaque tore passent deux fils d'écriture, l'un horizontal et l'autre vertical, et deux fils de lecture, l'un horizontal et l'autre vertical.

Si l'on veut écrire un 1 dans le tore E4, il suffira de sélectionner le fil de la colonne E et le fil de la rangée 4 (fils d'écriture) et de faire passer dans chacun de ces fils une impulsion capable de créer un champ H/2. Ce champ, à lui seul est insuffisant pour faire basculer un tore d'une position 0 (point B de la figure 6. 7) à une position proche de A.

Seul donc le tore de l'intersection E4 pourra basculer et inscrire dans sa mémoire le 1.

Pour lire ce même tore, on procèdera de même avec les 2 fils de lecture, E et 4, le champ créé dans les tores étant – H/2, seul le tore E4 basculera en position 0, créant une f.e.m. dans un fil de sortie qui, lui, est unique et traverse en diagonale tous les tores. S'il en sort une impulsion, c'est bien que le tore E4 était en position 1, puisque seul ce tore était en lecture à ce moment précis.

On remarquera même que ce fil de sortie traverse alternativement les tores dans un sens, puis dans l'autre si bien que toutes les petites impulsions générées par les tores qui ne sont pas en lecture (mais dont l'état se déplace légèrement sur la courbe d'hystérésis) ne s'additionnent pas entre elles mais au contraire ont tendance à s'annuler réciproquement.

Ces matrices de tores ont été à l'honneur au cœur des gros ordinateurs de gestion. Elles sont rapides, peu friandes en énergie et elles conservent les informations mises en mémoire même en cas de coupure de l'alimentation.

En revanche on constate que la lecture les remet automatiquement en position 0. On dit que la lecture est destructrice.

On peut remédier à ce défaut en restaurant le contenu de la mémoire tout de suite après sa lecture.

1.2.4 Adressage des tores d'une matric

En conservant l'exemple de la matrice d 64 tores, imaginons maintenant que l'on veuill sélectionner, pour l'écriture, le tore F3. L colonne F étant la sixième, elle aura le numéro puisqu'on commence la numération par 0.

Si l'on décide de commencer la sélection par l numéro de la colonne, suivi du numéro de la ligne l'*adresse* du tore F3 sera donc 5,3 c'est-à-dire, e binaire, 1 0 1.0 1 1.

Pour atteindre les fils d'écriture de ce tore faudra un dispositif du type de celui décrit a chapitre 2, § 5.4, c'est-à-dire un sélecteur d'adres se, ou plutôt ici un double sélecteur d'adresse (u pour les colonnes et un pour les lignes).

Le simple affichage de 1 0 1 dans le registre d sélecteur d'adresse des colonnes et de 0 1 1 dan celui des lignes mettra directement en relatio l'entrée commune avec les fils d'écriture F et (*fig.* 6. 11).

L'adresse d'une cellule de mémoire, dans un disposition en matrice, est donc la traduction e binaire du numéro de colonne et du numéro d ligne de cette cellule.

C'est en affichant cette adresse dans le registr de commande du sélecteur d'adresse que l'o sélectionne les fils d'écriture (ou de lecture correspondant à la cellule de mémoire.

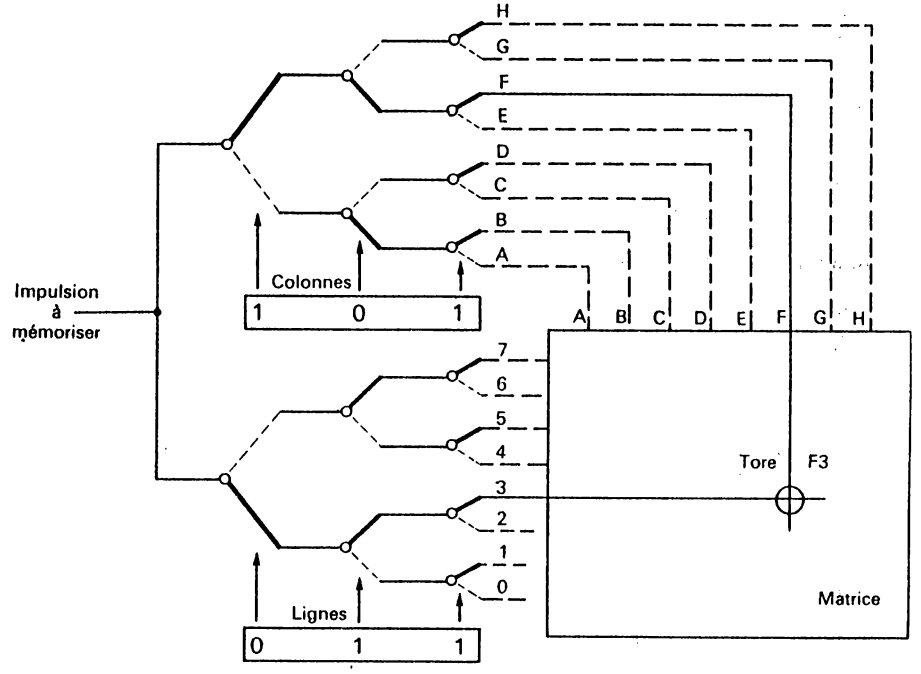


Fig. 6. 11

1.3 Technologies actuelles

Aujourd'hui de nouvelles technologies ont vu le jour, mais la structure matricielle et les systèmes d'adressage restent les mêmes.

1.3.1 Mémoires TTL

La cellule de mémoire TTL s'inspire du « flip-flop » c'est-à-dire de la bascule bistable que nous avons déjà étudiée. Des aménagements de cette bascule sont nécessaires pour en permettre une organisation en matrice.

Une cellule de mémoire est représentée sur la figure 6. 12. On y remarque en particulier deux fils de colonne, l'un à droite et l'autre à gauche, correspondant aux deux transistors de la bascule, ainsi qu'un fil de ligne.

Ces fils de ligne ou de colonne seront portés à des potentiels de 3 V pour la valeur 1, et de 0,5 V environ pour la valeur 0. On partira du principe que c'est le fil de colonne de gauche qui traduit la valeur mémorisée (le fil de colonne de droite indiquera alors son complément).

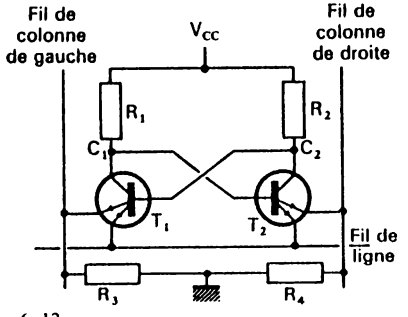


Fig. 6. 12

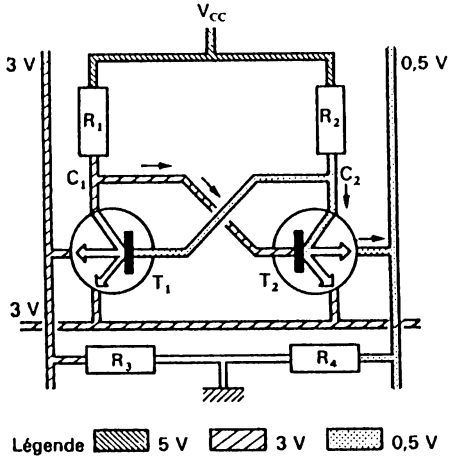


Fig. 6. 13

a) Écriture

Pour inscrire 1 dans une telle bascule nous allons donc porter le fil de ligne et le fil gauche de colonne au potentiel 3 V, et le fil de colonne de droite au potentiel 0,5 V (voir *fig.* 6. 13).

Un des 2 émetteurs de T_2 étant à 0,5 V, alors que sa base reçoit du courant à travers R_1, T_2 devient passant. Le potentiel de son collecteur (en C_2) est voisin de 0,5 V.

Comme les 2 émetteurs de T_1 sont portés à 3 V, alors que sa base (par C_2) est à 0,5 V, ce transistor T_1 est bloqué, ce qui augmente le potentiel en C_1 et sature T_2.

Cette situation est stable même après le passage des impulsions lorsque les 2 fils de colonne et le fil de ligne sont ramenés à un potentiel 0.

La valeur 1 a été enregistrée et elle est donc mémorisée sous forme d'une tension voisine de V_{CC} en C_1 et voisine de 0 en C_2.

b) Lecture

La lecture n'exige qu'une impulsion sur le fil de ligne. En effet :

— en l'absence de cette impulsion, le fil de ligne étant au potentiel 0 alors que le fil de colonne de droite est au potentiel du collecteur (environ 0,5 V), des 2 émetteurs de T_2, c'est l'émetteur E_1 branché sur le fil de ligne qui conduit (*fig.* 6. 14).

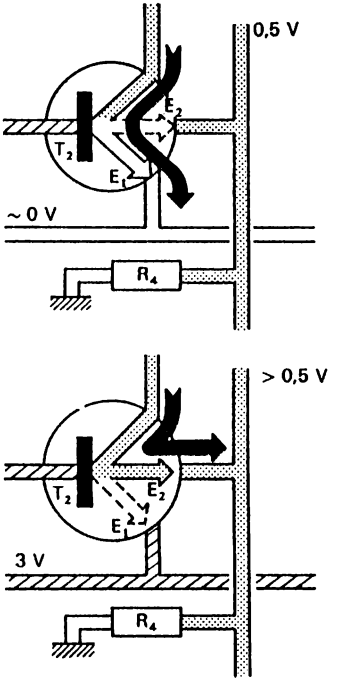


Fig. 6. 14

– Au passage de l'impulsion sur le fil de ligne l'émetteur E_1 ne conduit plus. E_2 devient alors conducteur. Le fil de colonne de droite est alors porté à un potentiel légèrement supérieur à 0,5 V, correspondant à cette tension de 0,5 V augmentée de la tension créée par le courant de E_2 à travers la résistance R_4 (ainsi que toutes les résistances analogues branchées en parallèle sur ce fil de colonne).

Un comparateur monté en sortie sur les 2 fils de colonne permet d'amplifier cette élévation de potentiel en sortie.

c) Bilan

La cellule de mémoire TTL présente un avantage essentiel qui est sa rapidité. Entre le moment où l'adresse est appliquée sur le registre sélecteur des lignes et des colonnes et le moment où l'information apparaît à la sortie (temps d'accès) il s'écoule environ 30 ns.

En contrepartie les cellules TTL sont consommatrices d'énergie, ce qui interdit une trop grande concentration de ces cellules à l'intérieur d'un circuit intégré, sous peine de le voir fondre.

Les recherches menées par les constructeurs aboutiront certainement à des circuits de plus en plus concentrés bénéficiant des performances des cellules TTL. Aujourd'hui on peut trouver des RAM de 1 024 cellules sous forme de boîtiers à 22 broches, tel le circuit MCM93422 de Motorola, dont le temps d'accès est de 30 ns et la dissipation thermique est de 0,26 milliwatt par bit).

1.3.2 Mémoires MOS statiques

Le schéma d'une cellule de mémoires MOS (*fig.* 6. 15) est très voisin de celui de la cellule TTL.

La principale qualité de ce type de mémoire est sa faible consommation qui autorise une forte intégration.

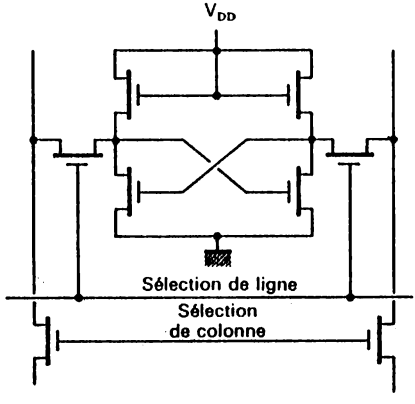


Fig. 6. 15

Pour diminuer encore cette consommation les constructeurs ont introduit un système de mise en veilleuse : lorsque le circuit n'est pas sollicité, les cellules sont placées sous faible tension (2 V), suffisante pour que l'information en mémoire soit préservée. Dès qu'un ordre de lecture ou d'écriture est appliqué, la tension de service (5 à 12 V) est aussitôt rétablie sur la colonne de la cellule activée.

On parvient ainsi à diviser par dix la consommation de la cellule au repos par rapport à celle de la cellule activée.

En contrepartie la mémoire MOS est lente. Le temps d'accès des premiers circuits fabriqués était de l'ordre de la microseconde. C'est pourquoi les constructeurs se sont-ils ingéniés à créer des variantes de toutes sortes, à réduire l'épaisseur et les dimensions des composants des MOS, à intégrer de véritables résistances miniatures de charge, etc. pour obtenir finalement des circuits-mémoires de performances assez remarquables, par exemple :

4 096 cellules intégrées dans le circuit MK4104 de Mostek, avec une consommation de 80 mW en activité et 8 mW au repos. Accès en 150 ns.

1 024 cellules dans un circuit AMI en VMOS, avec un temps d'accès de 50 ns.

1.3.3 Définitions

Toutes les mémoires que nous venons d'étudier sont appelées RAM (Random Access Memory) c'est-à-dire mémoires à accès aléatoire. Cela signifie que l'on peut accéder à chacune des mémoires séparément grâce à son adresse. Mais cette définition n'est pas complète : il est bon de préciser aussi que les RAM permettent d'enregistrer des informations, de les lire et de les effacer pour en enregistrer d'autres.

Une RAM perd les informations qu'elle contient dès que l'on coupe son alimentation. On dit que cette mémoire est « volatile ».

Les RAM organisées autour de bascules bistables (flips-flops) telles que celles qui ont été vues ici sont aussi appelées RAM *statiques* par opposition aux RAM *dynamiques* que nous étudions ci-dessous.

1.3.4 Les RAM dynamiques

Pour réaliser une RAM statique avec un flip-flop il faut intégrer 4 transistors MOS (dont 2 servent de charge). La recherche d'une diminution de ce nombre a abouti à la cellule à un seul MOS.

Dans ce cas, l'élément mémoire lui-même est constitué par la capacité drain-substrat de ce MOS. En beaucoup d'autres occasions on parlerait de capacité parasite, alors qu'ici elle est essentielle. Les fabricants l'accroissent, du reste, en modifiant les dimensions du MOS.

Le MOS étant donc adressé par le « gate », un 1 en écriture va charger cette capacité (*fig.* 6. 16).

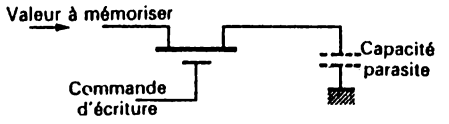


Fig. 6. 16

Comme le MOS est bidirectionnel, on pourra, par un nouvel adressage par le « gate », lire l'information.

Mais une capacité, parasite ou non, possède toujours une résistance de fuite, et se décharge à travers elle.

En quelques millisecondes la mémoire aura alors perdu son information.

Pour que la cellule conserve la valeur en mémoire, il faut restaurer cette valeur à intervalles réguliers. Cette opération s'appelle un « rafraichissement ».

Pour y parvenir on utilisera un dispositif dont le principe est celui de la figure 6. 17 : la mémoire est lue à travers un comparateur et cette lecture est aussitôt ré-introduite dans la cellule de mémoire.

L'entrée inverseuse du comparateur est portée à une valeur intermédiaire entre les tensions que donnerait une cellule chargée à 1 et une cellule chargée à 0. Ainsi le basculement de la sortie du comparateur est-il toujours sans équivoque.

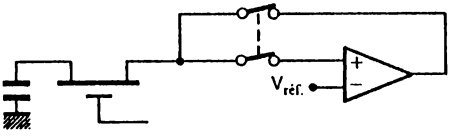


Fig. 6. 17

De telles mémoires qui peuvent être adressées, écrites et lues sont aussi des RAM. Comme ces mémoires « s'évaporent » on les a qualifiées, avec un certain humour, de mémoires dynamiques, ce qui signifie en fait qu'il faut les rafraîchir en général une fois toutes les deux millisecondes.

La nécessité de ce rafraîchissement complique quelque peu l'implantation de telles mémoires mais des constructeurs proposent des circuits intégrés contenant tout le dispositif de rafraîchissement. Dans d'autres cas c'est le microprocesseur auquel ces mémoires sont rattachées qui possède son propre dispositif de rafraîchissement (il s'agit du Z80).

1.3.5 Circuits et performances

La simplicité de conception et la faible consommation de la cellule MOS laissait entrevoir la possibilité d'une forte intégration de ces cellules dans un seul circuit. C'est effectivement ce qui s'est produit.

Les derniers circuits actuellement proposés par les constructeurs sont des mémoires de 2^{16} bits, c'est-à-dire 65 536 bits.

Citons par exemple le circuit EF6664 de Thomson-Efcis, ou encore le circuit 45H64 de Mostek.

En même temps les progrès technologiques ont permis de corriger le défaut majeur de ce type de cellules : son temps d'accès très long. Ainsi les temps d'accès des circuits qui viennent d'être cités sont-ils de l'ordre de 80 à 120 ns.

A ce degré d'intégration un nouveau problème se posait : comment proposer des mémoires adressables pour chacune des 65 536 cellules tout en maintenant le boîtier aux dimensions classiques des circuits mémoires (16 pattes). En effet, l'adresse seule exige l'envoi de 16 bits sur les entrées de sélection des colonnes et des lignes.

Pour tourner la difficulté, les constructeurs ont imaginé de faire parvenir cette adresse en deux temps : une première partie, la moitié basse composée des bits A_7 à A_0 est envoyée d'abord pour sélectionner les lignes ou rangées, puis, en un deuxième temps, c'est la partie haute, composée des bits A_{15} à A_8 qui parvient au circuit pour assurer la sélection des colonnes. Simultanément deux commandes sont activées tour à tour; il s'agit de RAS (Row Adress Strobe) pour les rangées et CAS (Column Adress Strobe) pour les colonnes (voir *fig.* 6. 18). Bien entendu une commande permet de placer le circuit en position lecture ou écriture ($R/\overline{W}$).

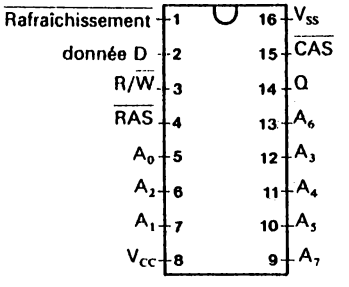


Fig. 6. 18 — Brochage de la mémoire EF 6664 (Doc. Thomson-Efcis)

1.4 Capacité des mémoires Unités de capacité

La bascule bistable unitaire est capable d'enregistrer 1 bit d'information.

Si nous disposons d'un registre de 4 bascules, nous pourrons les lire ou y enregistrer simultanément des informations, en les rangeant dans un ordre précis pour déterminer quels seront les bits de poids fort et de poids faible. Dans ce cas nous pourrons ranger ou lire un ensemble de 4 bits qu'on appelle souvent un *mot*.

La longueur des mots les plus courants est de 4, 8, 16 ou même 32 bits.

Par exemple, un circuit de 1 024 cellules peut être organisé en 1 024 × 1 bit, en 256 mots de 4 bits ou en 128 mots de 8 bits...

– Dans le premier cas, le circuit dispose de 10 broches d'adressage permettant de sélectionner chaque cellule de mémoire. Il n'aura qu'une seule entrée D et une seule sortie Q.

– Dans le deuxième cas (256 mots de 4 bits), les broches d'adressage seront au nombre de 8, et il faudra 4 entrées et 4 sorties (souvent confondues, et sélectionnées soit en entrée soit en sortie par la commande $R/\overline{W}$).

– Dans le troisième cas, il y aura 7 bornes d'adressage et 8 bornes de données.

Un mot de 8 bits est un standard du traitement de l'information (nous verrons plus loin pourquoi). On lui a donné un nom : c'est un *octet*.

Quant aux bits eux-mêmes, nous avons vu que l'on pouvait en stocker des quantités considérables dans les mémoires. On a donc créé une unité multiple du bit : le *kilo-bit*, symbolisé par la lettre K. Ce multiple n'obéit pas à la règle des nombres décimaux puisque les matrices de mémoires comportent un nombre de cellules égal à 2^n. Le kilo-bit correspond à 2^{10} bits ou 1 024 bits.

Les mémoires dynamiques vues plus haut, avec 65 536 cellules, peuvent donc contenir 64 K d'informations binaires.

Ces capacités suivront elles-mêmes la loi des puissances successives de 2 et on parlera de circuits de 1, 2, 4, 8, 16, 32, 64 K, etc.

1.5 Les mémoires mortes

Par opposition aux RAM statiques ou dynamiques qui viennent d'être étudiées, mémoires qui permettent à tout moment l'écriture ou la lecture, les mémoires dites *mortes* sont des mémoires dont le contenu ne peut être que lu et non modifié.

On appelle aussi ces mémoires des « ROM » (Read Only Memories) et non plus des RAM (Random Access Memories) alors qu'elles sont, tout autant que les RAM, à accès aléatoire c'est-à-dire, rappelons-le, qu'on pourra atteindre n'importe laquelle des informations qu'elles contiennent en appliquant l'adresse convenable sur les registres de sélection.

La ROM est donc une mémoire dont le contenu a été enregistré dès la fabrication, la plupart du temps en utilisant des masques correspondant aux souhaits de l'utilisateur. Ce caractère définitif entraîne quelques conséquences :

– Une ROM ne s'efface pas. Lorsqu'on coupe l'alimentation du circuit dans lequel elle est placée aucune des informations qu'elle contient ne se perd. Elle sera à nouveau capable de donner ces informations dès que l'alimentation sera rétablie.

– Chaque fois que l'on appellera, par son adresse, telle cellule de mémoire, la lecture donnera une information identique. Notamment, comme c'est le cas le plus fréquent si la ROM est organisée en mots de 4 ou 8 bits (*cf.* § 1.4) à chaque combinaison des bits d'adresse correspondra une combinaison précise des bits du mot lu.

On peut donc considérer une ROM comme un transcodeur.

1.5.1 Construction d'une ROM

On peut très bien construire une ROM sur un circuit imprimé double-face en utilisant des diodes et quelques amplis du type « buffer ».

Imaginons par exemple une matrice de 8 rangées de 4 cellules comme elle est représentée en figure 6. 19. Cette mémoire est organisée en mots de 4 bits (l'ensemble des sorties $Q_3\,Q_2\,Q_1$ et Q_0), chaque mot étant adressable par un nombre binaire à 3 bits et un sélecteur permettant d'envoyer une impulsion sur la ligne correspondante.

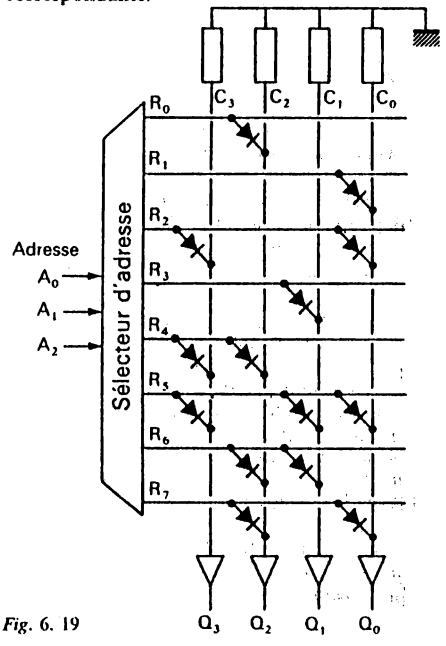


Fig. 6. 19

L'application de la valeur 101 (5_{10}) sur le registre du sélecteur envoie une impulsion sur la rangée R5. les diodes disposées sur cette rangée transmettent cette impulsion aux colonnes 0, 1 et 3 et le mot 1 0 1 1 apparaît en sortie.

En réalité les circuits sont établis avec des transistors bipolaires ou des transistors MOS, mais le principe est identique à celui des cellules à diodes.

1.5.2 Exemple d'application

La figure 6. 20 présente une application précise : celle d'un décodeur binaire pur-hexadécimal pour des afficheurs 7 segments. On y retrouvera la configuration exacte de la table de vérité établie pour l'exercice 1 du chapitre 4. Cette table de vérité est représentée sur la figure S. 4. 3 (voir solution des exercices).

Les sorties étant inversées, chaque diode correspond, en sortie, à un 0.

Cet exemple illustre bien le rôle de transcodeur d'une ROM. Nous étudierons plus tard le rôle très important qu'elles jouent dans le fonctionnement des microprocesseurs.

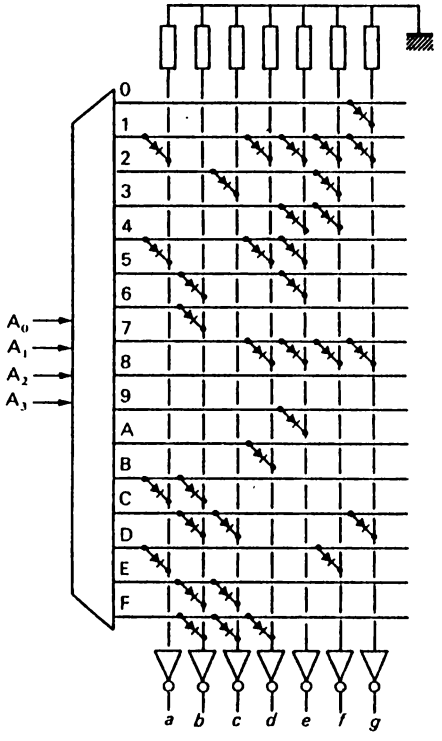


Fig. 6. 20

1.5.3 Les PROM

Les PROM sont des ROM « programmables », c'est-à-dire que le fabricant laisse à l'utilisateur le soin de définir et d'inscrire les informations à l'intérieur de ces mémoires.

Une fois cette inscription faite, la PROM ne pourra plus être effacée et ré-écrite; elle jouera exactement le rôle d'une ROM.

A chaque intersection d'une ligne et d'une colonne, le constructeur a placé un transistor prolongé par un fusible (*fig.* 6. 21). Ce fusible est capable de résister aux courants habituels des réseaux TTL (de l'ordre du milli-ampère), mais s'il est soumis à quelques dizaines de milli-ampères, il fond et provoque la coupure du circuit entre la ligne et la colonne, définissant un état 0 de la cellule correspondante.

Pour enregistrer ces informations dans la PROM il faudra utiliser un programmateur de PROM, permettant de sélectionner par leur adresse les cellules devant contenir un 0 et d'envoyer un courant de 20 à 30 mA dans les fusibles de ces cellules.

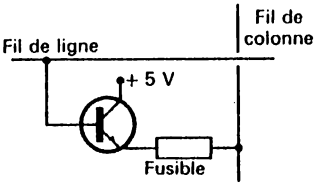


Fig. 6. 21

Remarque :

L'inscription de 0 par « claquage » du fusible est utilisée dans les PROM dont le fusible est en silicium polycristallin. Dans d'autres cas (fusibles à claquage de jonction), on envoie un courant sur des diodes en opposition de sorte qu'il y ait claquage de la diode placée en inverse. C'est alors un 1 que l'on inscrit en mémoire (*fig.* 6. 22).

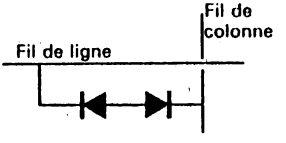


Fig. 6. 22

1.5.4 Les EPROM

Ayant obtenu les PROM, les constructeurs ont cherché à rendre celles-ci reprogrammables. L'invention d'un transistor particulier, le FAMOS a permis de trouver une solution à ce problème.

Le PMOS classique, représenté à la figure 6. 23, possède 3 connexions métalliques sur la source, le «gate» (ou grille) et le drain. L'application d'une différence de potentiel négative entre grille et source provoque l'apparition de charges négatives sur cette grille et de charges positives entre source et drain. Ces dernières vont permettre le franchissement de la barrière source/drain par le courant, grâce à l'éloignement des porteurs de charge négative dans le substrat. Le MOS devient passant.

Dès que la d.d.p. entre grille et source disparaît, le phénomène disparaît et le MOS se bloque.

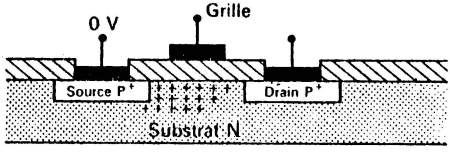


Fig. 6. 23

Le FAMOS (*fig.* 6. 24) se distingue de ce PMOS par le fait que la grille est emprisonnée dans la silice et qu'elle ne possède plus de fil de commande.

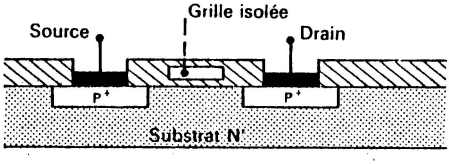


Fig. 6. 24

La source étant à la masse, si l'on applique au drain une tension fortement négative, il se produit un phénomène d'avalanche analogue à ce qui se produit dans une diode Zener polarisée en inverse.

Lorsqu'on annule alors cette différence de potentiel, on constate que le FAMOS est devenu passant. En effet, pendant le phénomène d'avalanche, des électrons, dans leur élan, ont franchi la mince couche isolante (0,1 micron) qui les séparait de la grille et se sont trouvés emprisonnés dans celle-ci.

Par effet capacitif, leur présence entraîne la formation d'un canal P dans le substrat, ce qui rend le FAMOS passant.

Cet état est stable et peut durer des années et même des siècles si l'on a fait durer le phénomène d'avalanche assez longtemps (de l'ordre de 30 secondes).

Pour que le FAMOS soit effaçable et reprogrammable il faut alors chasser les charges négatives emprisonnées dans sa grille. Pour cela il faut leur conférer une énergie qu'elles n'ont pas, en soumettant la grille à un rayonnement ultra-violet important pendant 15 à 20 minutes. Le FAMOS est alors à nouveau bloqué.

Pour laisser passer ce rayonnement, le circuit est muni d'une «fenêtre».

On a appelé ces mémoires des EPROM (Electrically Programmable Read only Memories) ou «ROM programmables électriquement».

On fabrique aujourd'hui des circuits EPROM contenant jusqu'à 16 K de cellules de mémoire. Une des plus courante est la mémoire 2816, contenant 16 K disposés en 2 K mots de 8 bits ou 2 K octets.

1.5.5 Les EAROM

Les mémoires RAM, on l'a vu, sont volatiles : dès que l'on coupe leur alimentation on perd en même temps les informations qu'elles contiennent.

Les ROM ne sont pas volatiles mais on ne peut que lire les informations qui y ont été enregistrées une fois pour toutes.

Avec les EPROM un pas est franchi entre la ROM et la RAM : la possibilité d'effacer puis de ré-inscrire d'autres informations. Il ne faut pas perdre de vue que lorsqu'on veut reprogrammer une EPROM il faut retirer cette mémoire du circuit dans laquelle elle est insérée, la placer sous le rayonnement ultra-violet qui va effacer tout son contenu, puis la placer dans un équipement spécial appelé programmateur d'EPROM.

L'idéal paraît être de disposer de ROM ayant toutes les qualités des RAM (mais on ne les appelerait plus ROM!) et qui conservent leur principale qualité : la non volatilité.

Un pas vers cet idéal a été franchi avec les EAROM (Electrically Alterable ROM).

Il s'agit de mémoires à FAMOS programmables et effaçables en quelques dizaines de millisecconde grâce à des impulsions électriques d'assez forte tension.

Dans l'avenir, les progrès permettront peut-être de faire de ces mémoires des substituts à beaucoup de celles que nous venons d'étudier.

1.6 Protection des sorties des circuits-mémoire

Lorsque l'on veut augmenter la capacité-mémoire d'un réseau à partir d'une configuration donnée, on se contente souvent d'augmenter le nombre de circuits-mémoire. Dès lors se posent quelques problèmes de protection des sorties.

1.6.1 Banalisation des lignes de sorties

Imaginons que, dans un réseau électronique, nous utilisions deux circuits de mémoires de 64 × 4 bits.

Chaque mot devant avoir son adresse, il nous faudra deux fois 64 adresses, soit 128 adresses différentes.

Le sélecteur d'adresse de chaque mémoire reçoit des adresses à 6 bits, de 000000 à 111111 (63), alors que l'adresse envoyée comportera 7 bits. Le bit de poids le plus fort servira donc à sélectionner l'un ou l'autre des deux circuits.

Pour cela, chaque circuit est muni d'une broche spéciale appelée CS (Chip Select) ou CE (Chip Enable). Cette commande étant activée permet le fonctionnement normal du circuit-mémoire.

En l'absence de la commande CE, le circuit est isolé de l'extérieur.

Grâce à ce dispositif il sera possible de brancher plusieurs circuits de mémoire sur une ligne banalisée d'information comme on le voit sur la figure 6. 25. Cette ligne de données est aussi appelée « bus des données ».

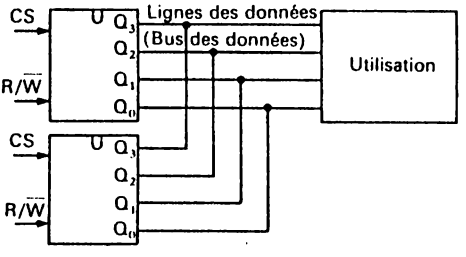


Fig. 6. 25

1.6.2 Portes 3 états

Pour isoler un circuit de la ligne de bus et éviter ainsi des courts-circuits entre des données incompatibles (un 1 et un 0) on dote les entrées/sorties des circuits-mémoire de portes « 3-états ». Elles sont appelées ainsi parce qu'elles peuvent effectivement fournir 3 états :

a) 1 état haut (donnée = 1)
b) 1 état bas (donnée = 0)
c) 1 état « haute-impédance », isolant pratiquement le circuit de l'extérieur.

En technolohie TTL on obtient ce résultat en organisant la sortie de la porte entre deux transistors mis en série entre V_{CC} et la masse (*fig.* 6. 26).

A l'état haut T_1 est passant et T_2 bloqué.

A l'état bas T_1 et T_2 sont passants.

A l'état haute impédance, T_1 et T_2 sont bloqués.

En technologie MOS, le troisième état est assuré par un transistor supplémentaire placé sur la ligne de sortie (*fig.* 6. 27).

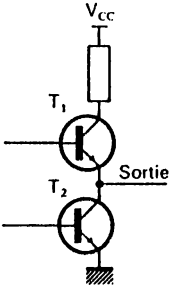


Fig. 6. 26

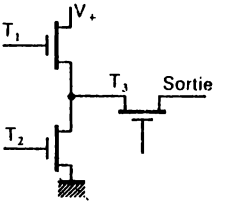


Fig. 6. 27

A l'état haut T_1 est passant, T_2 bloqué, T_3 passant.

A l'état bas T_1 est bloqué, T_2 passant, T_3 passant.

A l'état haute impédance, quels que soient les états de T_1 et T_2, T_3 est bloqué.

Ce principe peut être appliqué à n'importe quel type de porte (AND, NAND, OR, NOR, buffers ou inverseurs etc.).

La symbolisation de la commande du troisième état correspond, en normes américaines, aux schémas de la figure 6. 28. Cette commande est appelée E/D (Enable/Disable) et correspond à un niveau haut ou bas selon le gré des constructeurs. Il est donc important de consulter les notices.

En principe la présence d'un petit cercle sur la commande E/D indique que l'autorisation a lieu pour un niveau bas sur cette commande, l'absence de ce cercle, que l'autorisation est obtenu pour un niveau haut.

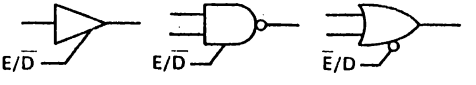


Fig. 6. 28

2. STRUCTURE INTERNE DES CIRCUITS MATRICIELS DE MÉMOIRES

L'étude de cette structure interne va se faire à propos d'un circuit RAM, comportant des circuits de lecture et d'écriture, la structure des ROM étant plus simple et se déduisant de la première par suppression des circuits d'écriture.

2.1 Circuits de lecture/écriture

On se souvient que l'écriture dans une cellule s'obtient par l'envoi d'une impulsion haute sur le fil de ligne et sur le fil de colonne gauche de cette cellule, alors que la lecture se fera par impulsion sur le fil de ligne et comparaison de la tension sur le fil de colonne droite avec une tension de référence (ou la tension sur le fil de colonne gauche) (voir notamment les *fig.* 6. 13 à 6. 15).

Les circuits de lecture/écriture seront associés aux colonnes de chaque matrice de cellules et comporteront (*fig.* 6. 29).

– Un ampli de lecture, chargé en général de comparer la tension sur le fil de colonne droite avec celle du fil de colonne gauche.

– Un registre tampon d'écriture, chargé de fournir les tensions d'écriture (niveau haut sur colonne gauche et niveau bas sur colonne droite dans le cas d'un 1 à écrire, et inversement pour un 0).

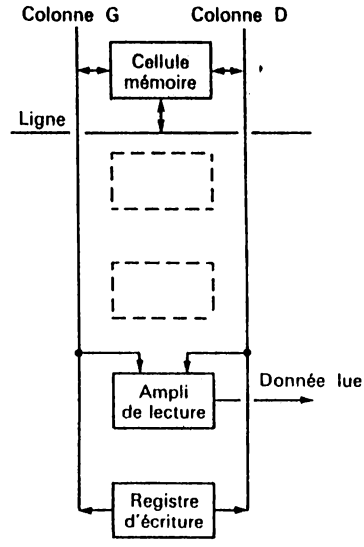


Fig. 6. 29

2.2 Circuits de contrôle des entrées/sorties

Pour que les amplis et registres précédents soient activés il faut qu'un organe, contrôlant les conditions imposées au circuit, fournisse les autorisations correspondantes (*fig.* 6. 30).

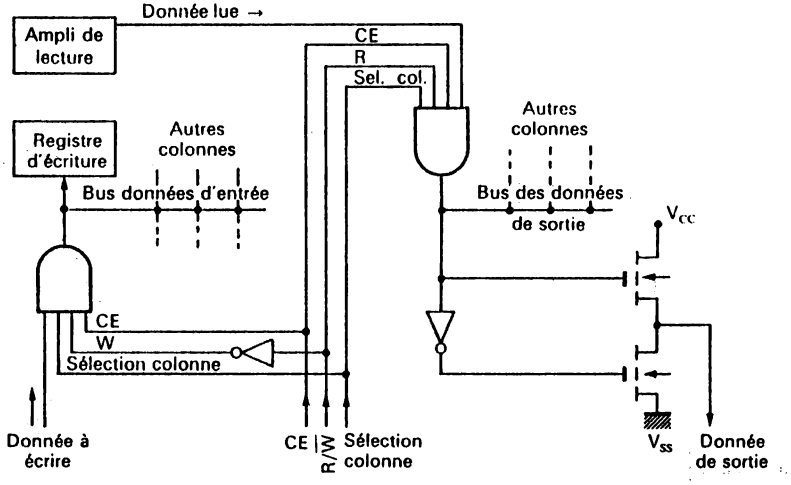


Fig. 6. 30

Ces conditions sont :

– CS ou CE (Chip Select ou Chip Enable) qui inhibe ou autorise le circuit entier.

– R/$\overline{W}$ (Read/Write) qui, selon les cas, active l'ampli de lecture ou le registre d'écriture.

– Data in ou Data out. Suivant les circuits ces deux informations figureront sur deux broches distinctes ou sur la même broche.

S'il s'agit de la même broche, c'est l'ordre R/$\overline{W}$ qui devra, par un jeu de portes « 3-états », instaurer un sens unique de circulation de cette information.

Enfin, à ces conditions seront associées les informations de sélection de colonne résultant de l'adressage.

2.3 Architecture d'ensemble

Elle découle des organes que nous venons d'étudier et qui sont raccordés à la structure matricielle des cellules de mémoires. Cette architecture d'ensemble figure sur la figure 6. 31.

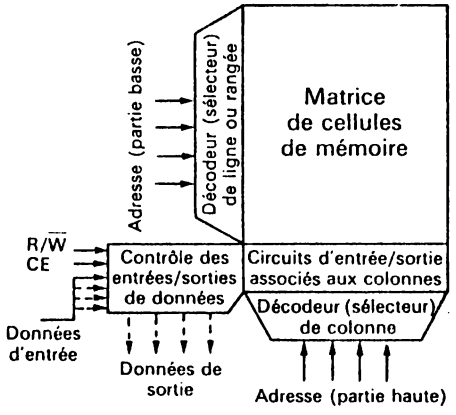


Fig. 6. 31

3. CHRONOGRAMMES DE LECTURE ET D'ÉCRITURE

Chaque fois qu'un constructeur propose sur le marché un circuit-mémoire, il joint à la documentation des chronogrammes d'utilisation de ce circuit.

Il est très important de savoir lire ces chronogrammes pour exploiter au mieux le circuit.

3.1 Chronogramme d'un cycle de lecture

Pour obtenir la lecture d'une cellule-mémoire (ou d'un groupe de cellules dans le cas où la mémoire est organisée en mots), il faut que les conditions suivantes soient réalisées :

– que l'adresse parvienne aux lignes et aux colonnes,

– que l'ordre « Chip Select » soit appliqué,

– éventuellement que l'autorisation « Chip Enable » soit donnée,

– que l'ordre « Read » soit appliqué,

– que le contenu de la mémoire soit prélevé en sortie.

Comme l'ordre « Chipt Select » provient de la partie haute de l'adresse, il sera appliqué en même temps qu'elle.

Notons encore que selon le type de mémoire nous aurons une borne Read ou une borne commune Read/Write. Dans ce dernier cas, c'est surtout CE qui va déclencher la lecture, sinon nous considérerons que l'ordre Read et la commande CE peuvent être appliqués en même temps (CE pouvant précéder Read).

Le plus important est d'appliquer l'adresse suffisamment tôt avant la lecture pour tenir compte du temps d'accès; ce temps comprend le décodage de l'adresse à travers le sélecteur (qui n'est pas inhibé par $\overline{CE}$), la lecture de la cellule, son enregistrement dans le comparateur de colonne, puis dans le « buffer » de sortie. Ce n'est qu'à ce dernier stade que la mémoire pourra être lue en sortie.

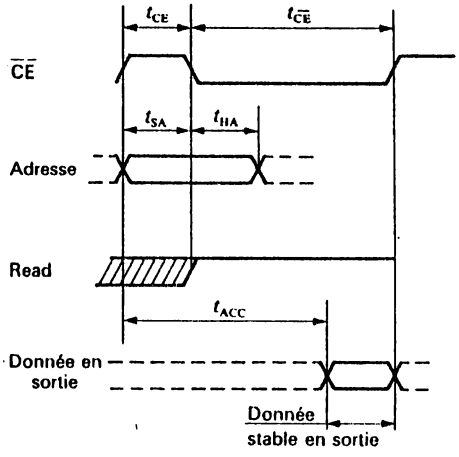
On jugera de l'importance de ce chronogramme lorsque les impulsions de commande sont de l'ordre de la microseconde et que le temps d'accès (pour certains MOS par exemple) est du même ordre.

La figure 6. 32 fournit un exemple de chronogramme appliqué à la mémoire MOS MM74C910.

Les abréviations ont les significations suivantes :

- t_{ACC} : Temps d'accès à partir de l'affichage de l'adresse (maximum 700 ns).
- t_{PD} : Délai de propagation de l'ordre « Read » aux sorties (540 ns).
- t_{SA} : Délai de décodage de l'adresse (minimum 160 ns).
- t_{HA} : Durée de maintien de l'adresse après décodage (minimum 20 ns).
- $t_{\overline{CE}}$: Durée de l'impulsion « $\overline{\text{Chip Enable}}$ » négative (minimum 600 ns).
- t_{CE} : Durée de l'impulsion CE d'inhibition, nécessaire entre deux adressages successifs (minimum 260 ns).

La zone hachurée du créneau « Read » indique que cet ordre peut être déclenché à un moment



Cycle de lecture

Fig. 6. 32

quelconque à l'intérieur de cette zone, sans inconvénient pour la suite des opérations.

On constatera surtout que le moment propice à la lecture (zone «donnée stable en sortie») intervient alors que l'adresse peut ne plus être appliquée depuis un certain temps, de l'ordre de 500 ns pour cette mémoire.

3.2 Chronogramme d'un cycle d'écriture

Sur le même principe que pour la lecture, les constructeurs fournissent des chronogrammes pour l'écriture en mémoire.

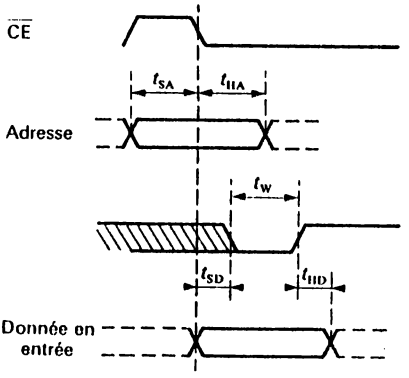
La figure 6. 33 donne le chronogramme d'écriture de la mémoire MMC74C910.

Outre les abréviations précédemment explicitées, on trouvera :

t_{SD} : délai de transfert de la donnée d'entrée (nul ici),

t_{HD} : temps minimum de maintien de la donnée d'entrée (50 ns),

$t_{\overline{W}}$: durée minimum de l'impulsion de lecture (180 ns).



Cycle d'écriture

Fig. 6. 33

On notera surtout que, dans le cas de l'écriture, la donnée doit être présente à l'entrée dès que l'adresse est décodée et que CE «autorise» le circuit.

EXERCICES

1. On dispose, sur un circuit, d'une « unité de calcul et de gestion » dotée d'un bus d'adresses de 10 bits et d'un bus de données de 8 bits.

On se propose d'équiper ce circuit de mémoires RAM. Les circuits choisis sont des matrices de 32 × 32 cellules, où chaque cellule est adressable individuellement (1 024 « mots » de 1 bit). Ces mémoires possèdent en outre :

— une commande d'écriture-lecture ($R/\overline{W}$),

— une commande d'autorisation d'entrée/sortie ($\overline{CE}$).

On veut, grâce à l'adressage sur 10 bits, pouvoir lire ou écrire des mots de 8 bits (octets).

Comment organiser ces circuits de RAM et aiguiller les informations du bus de données dans un sens ou dans l'autre suivant que l'on sera en lecture ou en écriture?

2. Exercice guidé : Réalisation d'un répertoire téléphonique.

Nous vous proposons de réaliser un répertoire téléphonique gardant en mémoire 256 numéros de téléphone (on supposera qu'il s'agit de numéros à 7 chiffres, mais ce répertoire est extensible à 8 chiffres par exemple).

1° Circuits de mémoires

La mémoire de «base» sera le circuit TMS 4043, dont les caractéristiques sont les suivantes :

– Alimentation entre 5 V et masse.

– Mémoire MOS mais entièrement compatible avec les circuits TTL.

– Temps d'accès de 650 ns.

– Adressage sur 8 bits.

– Organisation en 256 mots de 4 bits (1 024 cellules organisées en une matrice de 32 × 32 cellules).

– Entrées et sorties sont sur les mêmes broches. L'ordre $\overline{W}$ met en haute impédance les sorties grâce à des portes «trois-états», et permet l'entrée des données à écrire.

Le brochage de ce circuit est explicité sur la figure E.6. 1 et son schéma interne sur la figure E. 6. 2.

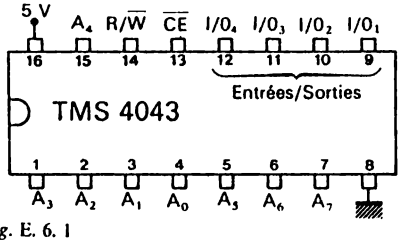


Fig. E. 6. 1

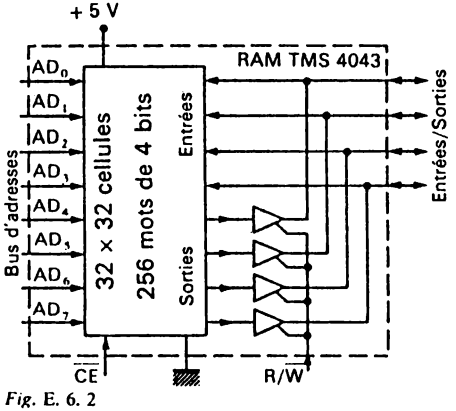


Fig. E. 6. 2

2° Extension des mots de mémoire

Chaque mémoire va permettre d'enregistrer un chiffre. Pour sept chiffres il faudra donc sept circuits.

Ces sept chiffres correspondant tous à un même abonné, le même bus d'adresse desservira simultanément tous les circuits de mémoires.

A chacun de ces circuits de mémoire sera associé un décodeur et un afficheur sept segments, ce qui permettra de lire le numéro (on peut aussi imaginer un système de multiplexage-démultiplexage comme cela a déjà été vu).

3° Alimentation

Nous avons affaire à une RAM. Elle doit être alimentée en permanence sous peine de perdre son contenu. Il faudra prévoir une alimentation par pile ou accumulateur.

4° Principe d'organisation du répertoire

– Nous disposons de 256 adresses binaires.

– Le répertoire doit permettre de trouver le numéro de téléphone d'une personne dont nous connaissons le nom.

– C'est donc à partir du nom, écrit en clair sur une fiche, que nous devons pouvoir lire le numéro mis en mémoire.

A ce numéro correspondra une des 256 adresses du bus d'adresses.

Il faut donc affecter une adresse binaire à chaque nom, rendre cet adressage automatique par le seul choix de la ligne correspondant au nom.

Il faudra ensuite écrire le numéro de téléphone correspondant en mémoire.

Enfin, on organisera le circuit de telle sorte qu'on puisse soit lire, soit écrire (modifications, créations de nouvelles lignes, etc.).

5° Création du fichier des noms

Il est illusoire de vouloir aligner 256 noms en une seule fiche. On disposera plutôt d'une dizaine de fiches de 25 à 26 noms disposés par ordre alphabétique si possible.

Il n'est pas nécessaire d'affecter les adresses binaires, pour chaque nom, dans un ordre lié à la place du nom sur les fiches (ce serait se créer des problèmes insolubles lors des modifications et remaniements de fichier).

On disposera donc, par ailleurs, d'un carnet comportant les numéros de 0 à 255, et, au fur et à mesure de l'inscription d'un nom sur les fiches, on affectera à ce nom le *premier* numéro libre.

Il reste à constituer l'adressage automatique.

6° Adressage des mémoires par le nom

Chaque fiche sera organisée en mémoire morte.

On peut, pour cela, utiliser le système des diodes réunissant les lignes aux colonnes pour inscrire des 1.

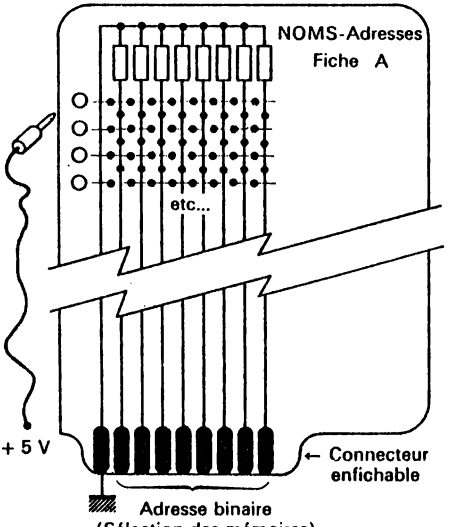


Fig. E. 6. 3

Ainsi une fiche sera réalisée grâce à un circuit double face comportant des colonnes (sur la face avant par exemple) aboutissant à un connecteur enfichable sur le bus d'adresses et, sur l'autre face, des lignes reliables aux colonnes par des diodes, à l'extrémité de chaque ligne une prise dans laquelle on pourra ensérer une petite prise banane reliée aux 5 volts permettra la sélection de l'adresse.

Sur le côté face on pourra inscrire les noms des abonnés, ainsi que leur «adresse» véritable, (*fig.* E. 6. 3).

Il restera alors à donner les diodes correspondant aux 1 de l'adresse binaire affectée à chaque abonné.

7° Écriture du numéro de téléphone dans les mémoires

Ayant sélectionné un nom grâce à la ligne sur laquelle il est inscrit dans la fiche alphabétique électronique, son adresse binaire parvient à tous les circuits mémoires. Il faut alors pouvoir accéder à chaque bus d'information de 4 fils pour faire parvenir un des chiffres correspondant au numéro de téléphone.

On peut réaliser une carte «mémoire morte» analogue aux fiches alphabétiques précédentes. Cette carte unique est dotée de quatre colonnes et de 10 lignes codant les chiffres binaires de 0 à 9.

Pour chaque chiffre à inscrire on sélectionnera la ligne par une fiche banane amenant la tension de 5 volts. On connectera cette fiche au bus de données de la mémoire correspondante puis on mettra le $\overline{CE}$ général et l'ordre R/$\overline{W}$ de la mémoire simultanément au niveau bas.

Si l'on relâche alors R/$\overline{W}$, les données enregistrées vont se présenter sur le bus. Elles ne peuvent entrer en conflit avec les informations portées par la carte puisque ce sont les mêmes.

Pourtant, s'il s'agit de modifier un chiffre déjà entré en mémoire, on risque des courts-circuits au moment où l'on connecte cette carte si l'ordre W n'a pas été envoyé. Il sera donc prudent de compléter la carte par des portes trois-états libérant les valeurs binaires du chiffre dans le seul cas ou R/$\overline{W}$ est à 0 (*fig.* E. 6. 4).

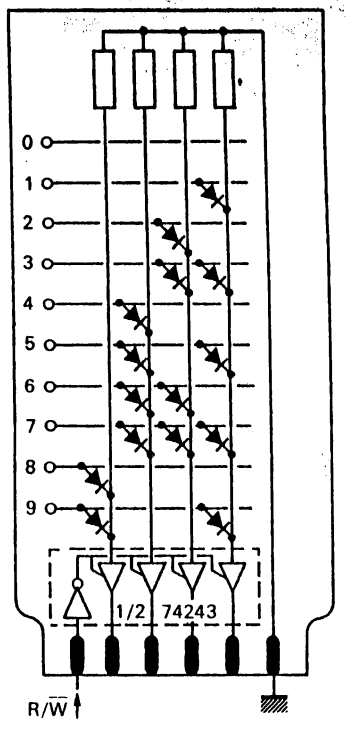


Fig. E. 6.4

Le schéma d'ensemble du circuit «répertoire téléphonique» est donné en figure E. 6. 5.

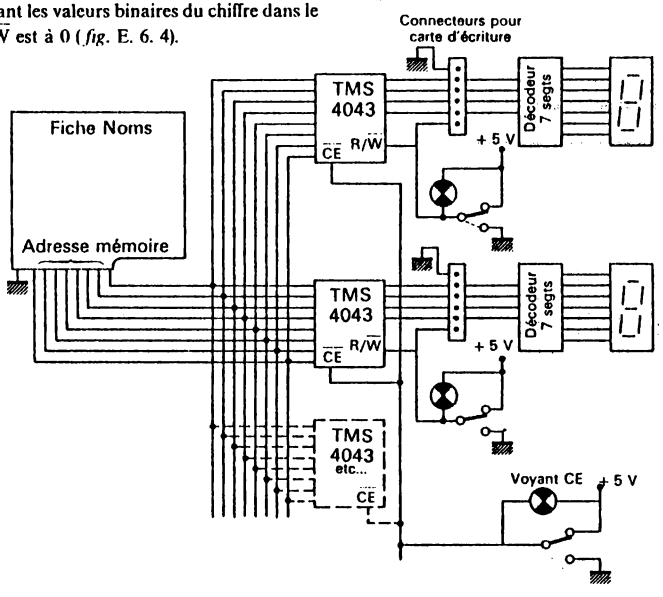


Fig. E. 6. 5

CHAPITRE 7

Les opérations de calcul arithmétique

INTRODUCTION

Nous avons traité, dans les premiers chapitres de cet ouvrage, des opérations logiques (AND, NAND, OR, etc.) que l'on pouvait effectuer avec deux ou plusieurs variables binaires.

Nous allons voir maintenant comment réaliser les opérations arithmétiques telles que l'addition, la soustraction et la multiplication (nous ne traiterons pas ici de la division).

1. L'ADDITION

Considérons deux nombres binaires purs à additionner. Ces deux nombres (de 4 bits chacun, par exemple) sont rangés dans deux registres, A et B.

Comme dans l'addition de nombres décimaux, nous allons procéder par additions élémentaires, en commençant par les valeurs a_0 et b_0 de poids le plus faible (unités) puis les valeurs de poids 2, 4 et 8.

Quelles que soient les valeurs que nous trouverons, elles ne peuvent être que 0 ou 1, si bien que la table d'addition binaire se résume au tableau suivant

0 plus 0 = 0
0 plus 1 = 1
1 plus 0 = 1
1 plus 1 = 10 (0 et 1 de retenue)

Notez que nous écrivons « plus » et non pas « + » pour ne pas introduire de confusion entre l'addition (opération arithmétique) et le OU (opération logique).

En fait les trois premières lignes de ce tableau sont identiques à celles de l'opération logique $a_0 + b_0$. Seule la dernière diffère, et si l'on ne tient compte que de la valeur des unités du résultat on obtient alors la table de vérité de l'opération logique OU EXCLUSIF :

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

Cette constatation va nous mettre sur la voie d'une solution pour la réalisation d'un circuit additionneur.

1.1 Le demi-additionneur

Comme il peut y avoir une retenue de 1 dans l'addition de deux valeurs binaires, il est évident qu'un seul fil de sortie du résultat ne suffit pas. Il faudra deux fils, le premier (S) donnera le résultat « direct », le second donnera la valeur de la retenue R. Dès lors, les valeurs que nous devons avoir sur chacun de ces fils sont les suivantes :

a_0 b_0	S	R
0 plus 0 →	0	0
0 plus 1 →	1	0
1 plus 0 →	1	0
1 plus 1 →	0	1

On remarque lors que si S se déduit de a_0 et b_0 par la relation EXOR, R résulte d'une relation ET.

D'où la réalisation du circuit d'addition de a_0 plus b_0 (*fig.* 7. 1).

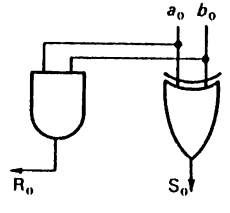


Fig. 7. 1

Ce schéma correspond à ce qu'on appelle un demi-additionneur. Il répond bien en effet au problème posé par l'addition des deux valeurs unités de A et B. Mais si nous voulons maintenant additionner les deux valeurs de rang supérieur, a_1 et b_1, il va falloir tenir compte de la retenue de a_0 plus b_0.

1.2 Le module d'addition binaire

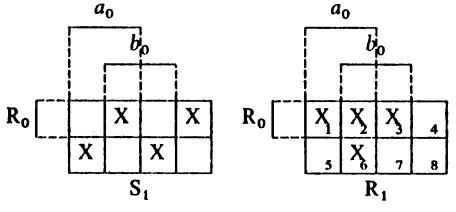
Le module complet d'addition va donc disposer de 3 entrées pour a_1, b_1 et la retenue de a_0 plus b_0 (R_0).

Il aura par ailleurs 2 sorties pour S_1 et R_1.

Établissons la table de vérité liant S_1 et R_1 à a_1, b_1 et R_0.

a_1	b_1	R_0	S_1	R_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A partir de cette table de vérité nous pouvons établir les tableaux de Karnaugh pour S_1 et R_1 :



Ce qui permet d'écrire les relations logiques suivantes :

$$S_1 = R_0 a_0 b_0 + R_0 \bar{a}_0 \bar{b}_0 + \bar{R}_0 \bar{a}_0 b_0 + \bar{R}_0 a_0 \bar{b}_0$$
$$= R_0(a_0 b_0 + \bar{a}_0 \bar{b}_0) + \bar{R}_0(\bar{a}_0 b_0 + a_0 \bar{b}_0)$$
$$S_1 = R_0\overline{(a_0 \oplus b_0)} + \bar{R}_0(a_0 \oplus b_0)$$

On constate que S_1 est la fonction OU EXCLUSIF de R_0 et $a_0 \oplus b_0$

$$\boxed{S_1 = R_0 \oplus (a_0 \oplus b_0)}$$

Quant à R_1 on peut le formuler comme un OU logique entre les cases 1 et 3 et les cases 2 et 6 :

$$R_1 = R_0(a_0 \bar{b}_0 + \bar{a}_0 b_0) + a_0 b_0$$

$$\boxed{R_1 = R_0(a_0 \oplus b_0) + a_0 b_0}$$

A partir de ces deux formulations, il est facile de construire un étage complet d'additionneur (cf. *fig.* 7.2).

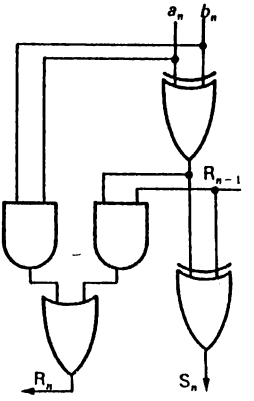


Fig. 7. 2

1.3 L'additionneur à N étages

La réalisation d'un additionneur à N étages consiste à mettre bout à bout N modules du type précédent (*fig.* 7. 2), reliés par le fil de retenue de l'étage précédent.

Pour le premier étage il suffira de mettre à la masse le fil de retenue d'entrée.

On peut ainsi imaginer la réalisation d'un additionneur à 4, 8, 16 bits pour chaque nombre à additionner.

Seul un inconvénient limitera le nombre de modules : le temps de réponse.

En effet, le module de base que nous venons d'étudier est très rapide : on peut espérer une réponse en 25 à 50 ns à l'addition de *a* et *b*. Nous avons vu que l'utilisation de portes de logique négative (NOR, NAND) permettait d'accélérer cette réponse. Mais, si l'on relie 16 modules entre eux, la réponse ne sera donnée que lorsque toutes les retenues auront été «comptabilisées». Ces retenues se propageant d'un étage à un autre, il faudra $16 \times 40 = 640$ nanosecondes pour l'obtention d'un résultat correct. Ce délai peut être déjà qualifié de long dans les applications de traitement de l'information.

C'est pourquoi on a recours, pour raccourcir le temps de réponse, à un procédé qui permet d'anticiper la retenue pour un étage en fonction des valeurs qui apparaissent sur les étages précédents.

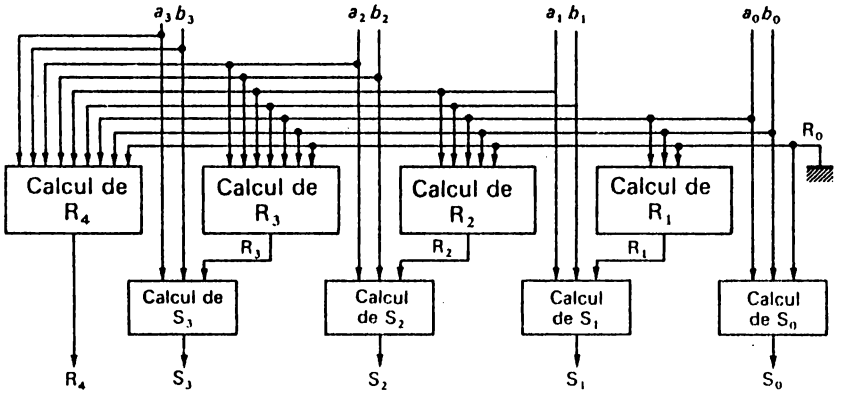


Fig. 7. 3

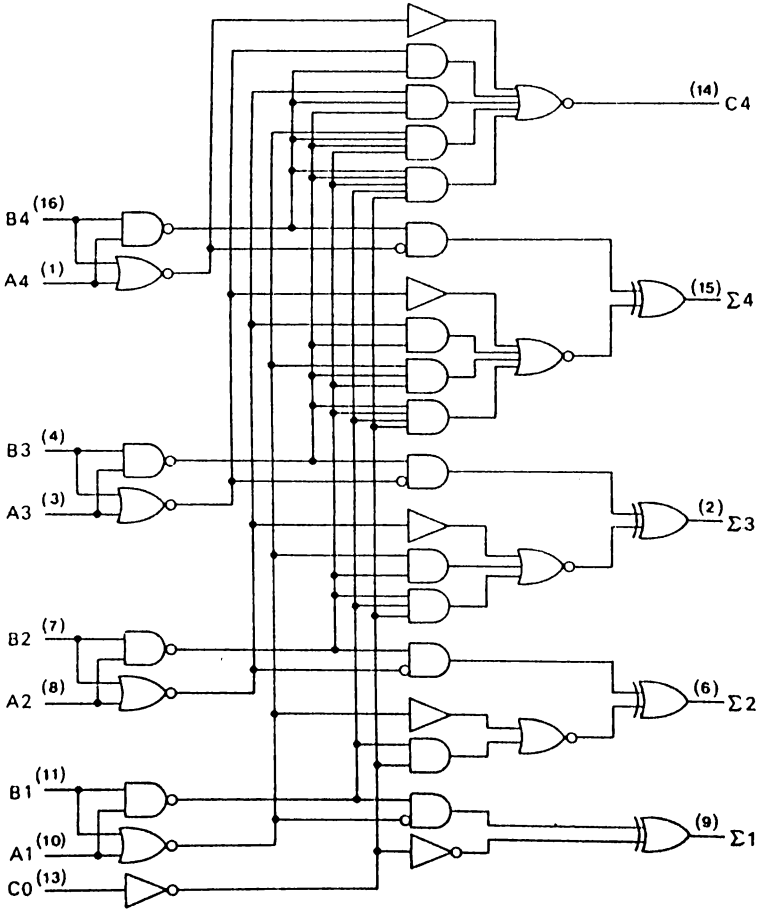


Fig. 7. 4 (Doc. Texas Instruments)

Le principe de ce système est celui de la figure 7. 3. On y voit que pour un étage donné, la retenue qui doit être prise en compte est calculée au *niveau de cet étage* par un circuit de calcul qui prend en compte toutes les valeurs précédentes et la retenue initiale. Par ce stratagème, même si on allonge le temps de réponse de l'étage, ce temps ne s'additionne pas avec le temps de réponse des étages précédents. Le temps global est égal au temps de réponse de l'étage le plus lent.

C'est sur ce principe qu'est construit le circuit 7483, additionneur de nombres de 4 bits qui donne le résultat en moins de 25 nanosecondes (voir *fig.* 7. 4).

Ici la retenue initiale est appelée C_0. Les nombres sont $A_4A_3A_2A_1$ et $B_4B_3B_2B_1$ (poids faible à droite), le résultat est donné par $C_4\Sigma_4\Sigma_3\Sigma_2\Sigma_1$, C_4 étant la dernière retenue de l'addition.

On pourra, bien sûr, mettre en cascade plusieurs circuits 7483, mais cette fois-ci il faudra additionner le temps de réponse de chaque circuit pour avoir le temps de réponse total.

2. LA SOUSTRACTION

La soustraction, en binaire, est très proche de l'addition grâce à un petit stratagème très simple. Mais voyons déjà comment on peut remplacer une soustraction par une addition dans notre système décimal.

Imaginons que nous ayons à faire la soustraction 215 – 172. Nous pouvons transformer 172 en remplaçant chaque chiffre par leur complément à 9. 172 devient alors 827. Si nous additionnons alors 215 et 827 nous obtenons 1 042. Il suffira de supprimer la dernière retenue à gauche et à ajouter 1 pour obtenir le résultat de la soustraction : 43. Ceci ne tient pas d'un tour de passe-passe. En prenant le complément à 9 de chaque chiffre de 172, nous avons obtenu le nombre 999 – 172.

L'addition a donc consisté à chercher le résultat de

$$215 + 999 - 172 = 1\,000 + (215 - 172) - 1 = S$$

Le bon résultat de la soustraction s'obtiendra bien en faisant S – 1 000 + 1.

La soustraction binaire peut utiliser le même stratagème, avec plus de facilité encore, car on prendra, non plus le complément à 9, mais le complément à 1 des chiffres du nombre à soustraire.

En binaire 215 s'écrit 1 1 0 1 0 1 1 1
et 172 1 0 1 0 1 1 0 0
le complément à 1 des chiffres de 172 donnera donc le nombre 0 1 0 1 0 0 1 1, et il suffira de faire l'addition

$$\begin{array}{r} 11010111 \\ +\,01010011 \\ \hline (1)00101010 \end{array}$$

En supprimant la retenue de gauche et en ajoutant 1 à droite, on obtient le nombre

101011

qui correspond bien au nombre décimal 43.

2.1 Réalisation d'un soustracteur

Puisqu'on a ainsi ramené la soustraction à une addition, on ne construira pas un soustracteur, mais on fera en sorte de disposer à la fois d'un additionneur et d'un soustracteur.

Il faudra, pour cela, deux commandes séparées; l'une pour l'addition et l'autre pour la soustraction.

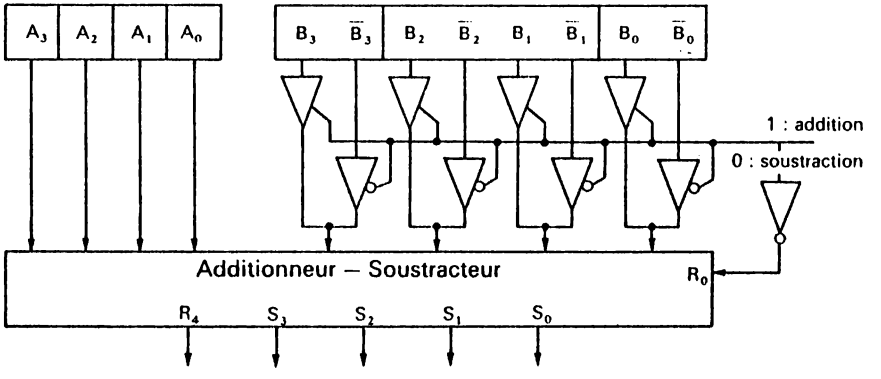


Fig. 7. 5

La commande de soustraction devra sélectionner les compléments à un des chiffres du nombre B.

Il suffit pour cela que le registre qui renferme ce nombre B soit doté de sorties Q et $\overline{Q}$. La commande soustraction (−) autorisant les sorties $\overline{Q}$ sur les entrées $b_3 b_2 b_1$ et b_0 grâce à des portes « 3 états » (*fig.* 7.5).

De plus cette même commande « − » doit mettre la retenue initiale (R_0) à la valeur 1 alors que la commande d'addition doit la mettre à 0.

Enfin la lecture du résultat ne doit pas prendre en compte le dernier bit de retenue. Notez que ce bit doit être à 1. S'il est égal à 0, cela signifie que le résultat de la soustraction est négatif. Dans ce cas le résultat est le complément à 2^N du nombre négatif, N étant le nombre d'étages de l'additionneur.

Par exemple, l'opération 172 − 215 donnera :

```
  1 0 1 0 1 1 0 0
+ 0 0 1 0 1 0 0 0
-----------------
0 1 1 0 1 0 1 0 0
        +     1
-----------------
```

0 1 1 0 1 0 1 0 1 = 213 (256 − 213 = 43)

3. LA COMPARAISON

A partir du soustracteur il est relativement aisé de concevoir un comparateur qui indiquera sur 3 fils de sortie soit A < B, soit A = B, soit A > B.

Dans le premier cas en effet la retenue de gauche sera à 0, dans le dernier cas elle sera à 1. Enfin si A = B la soustraction de tous les bits de A et de B ne donnera que des 1 (et 0 en retenue). Il suffira de mettre en évidence cette concordance de 1 par un système de portes pour émettre 1 sur le fil A = B.

4. UNITÉ DE CALCUL ARITHMÉTIQUE ET LOGIQUE

Les constructeurs se sont attachés à réaliser sur une seule « puce » un circuit capable à la fois de faire les opérations arithmétiques telles que l'addition, la soustraction et la comparaison et aussi les opérations logiques telles que AND, NAND, OR, NOR, EXOR et une dizaine d'autres. C'est le circuit TTL 74181 (*fig.* 7. 6).

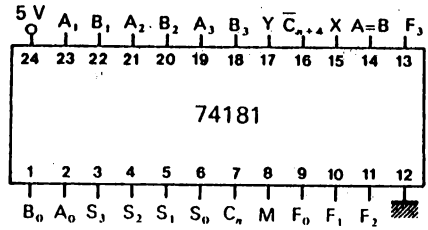


Fig. 7. 6

4.1 Les entrées du circuit 74181

Ce circuit ayant, entre autres fonctions, celle d'additionner ou de soustraire, il est doté de 4 entrées pour le nombre A et 4 autres pour le nombre B. Ce sont les entrées $A_0 A_1 A_2 A_3$ et B_0, B_1, B_2, B_3.

En outre nous trouvons une entrée pour la retenue de poids faible appelée C_n. Rappelons que $C/_n$ doit être à 0 pour les additions et à 1 pour les soustractions.

4.2 Les sorties du circuit 74181

Nous trouvons tout d'abord les 4 sorties $F_0 F_1 F_2 F_3$ qui correspondent aux 4 bits de résultat des différentes fonctions assurées.

La retenue de poids fort est donnée sur la broche $C/_{n+4}$.

Une sortie A = B indique l'égalité des deux nombres A et B. Attention, cette sortie est à collecteur ouvert. Il faudra donc placer une résistance entre l'alimentation et cette borne. L'intérêt de ce collecteur ouvert est de permettre de réaliser très simplement un AND « fantôme » avec plusieurs circuits mis en cascade (nombres de 8 ou 16 bits à comparer).

Enfin deux sorties, X et Y sont destinées à la liaison avec un circuit 74182, générateur anticipant des retenues pour quatre circuits 74181.

4.3 Les commandes du 74181

La commande M à l'état haut permet de réaliser les fonctions logiques. M = 0 transforme le circuit en calculateur arithmétique (avec quelques fonctions logiques).

Quatre fils de sélection, S_0, S_1, S_2 et S_3 permettent dans chaque cas (M = 1 ou M = 0) 16 combinaisons différentes, ce qui rend donc le circuit capable de 32 fonctions différentes, représentées sur le tableau ci-après :

Sélection S_3 S_2 S_1 S_0	M = 1 Opérations logiques	M = 0 Opérations arithmétiques	
		$\overline{C}_n = 1$ (pas de retenue)	$\overline{C}_n = 0$ (il y a une retenue)
0 0 0 0	$F = \overline{A}$	F = A	F = A plus 1
0 0 0 1	$F = \overline{A + B}$	$F = A + \overline{B}$	F = (A + B) plus 1
0 0 1 0	$F = \overline{A}B$	$F = A + \overline{B}$	F = (A + $\overline{B}$) plus 1
0 0 1 1	F = 0	F = moins 1 (complém^t à 2)	F = 0
0 1 0 0	$F = \overline{AB}$	F = A plus $A\overline{B}$	F = A + $A\overline{B}$ plus 1
0 1 0 1	$F = \overline{B}$	F = (A + B) plus $A\overline{B}$	F = (A + B) plus $A\overline{B}$ plus 1
0 1 1 0	$F = A \oplus B$	F = A moins B moins 1	F = A moins B
0 1 1 1	$F = A\overline{B}$	F = $A\overline{B}$ moins 1	F = $A\overline{B}$
1 0 0 0	$F = \overline{A} + B$	F = A plus AB	F = A plus AB plus 1
1 0 0 1	$F = \overline{A \oplus B}$	F = A plus B	F = A plus B plus 1
1 0 1 0	F = B	F = (A + $\overline{B}$) plus AB	F = (A + $\overline{B}$) plus AB plus 1
1 0 1 1	F = AB	F = AB moins 1	F = AB
1 1 0 0	F = 1	F = A plus A	F = A plus A plus 1
1 1 0 1	$F = A + \overline{B}$	F = (A + $\overline{B}$) plus A	F = (A + $\overline{B}$) plus A plus 1
1 1 1 0	F = A + B	F = (A + $\overline{B}$) plus A	F = (A + $\overline{B}$) plus A plus 1
1 1 1 1	F = A	F = A moins 1	F = A

N.B. : Le signe + correspond au OU logique.

5. LA MULTIPLICATION DE NOMBRES BINAIRES

Pour imaginer un « multiplieur » binaire il faut analyser les lois d'obtention du résultat à partir des valeurs à multiplier.

La multiplication en décimal procède de trois opérations :

a) la multiplication chiffre par chiffre qui nécessite la connaissance de la « table de multiplication »,

b) le décalage des résultats successifs correspondant aux unités, dizaines, centaines, etc.,

c) l'addition des résultats partiels.

Nous allons transposer ceci au langage binaire.

5.1 La table de multiplication binaire

Beaucoup plus simple que la table de multiplication décimale, la table de multiplication binaire se résume à

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Cette table est identique à celle de la fonction ET (AND) ou produit binaire, ce qui justifie le symbole « . » choisi pour cette relation logique.

La porte AND classique permet donc d'obtenir le produit de 2 valeurs binaires *a* et *b*.

5.2 Réalisation d'un multiplieur de deux nombres de 3 bits

En s'en tenant à la structure même des opérations de multiplication on peut imaginer le schéma d'un circuit permettant de multiplier A et B, chacun de ces nombres ayant 3 bits. L'opération comportant 2 décalages et 1 retenue possible, il faudra afficher le résultat sur 6 sorties (S_0 à S_5). On obtient alors le schéma de la figure 7. 7.

La seule différence entre la méthode utilisée et la méthode classique réside dans le fait que les additions se font au coup par coup au lieu d'être faites globalement à la fin, puisque les modules d'addition ne prennent en compte que deux valeurs à additionner.

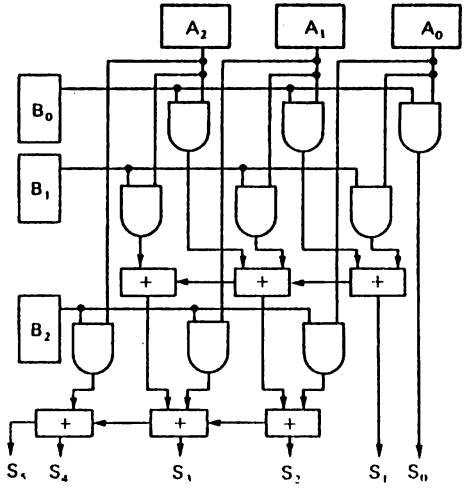


Fig. 7. 7

5.3 Multiplieur en circuit intégré

Le circuit TTL 74274, basé sur des principes proches de l'exemple précédent, contient un circuit multiplieur de deux nombres binaires de 4 bits. Il possède donc 8 sorties, protégées par des portes «trois-états» commandées par les broches $\overline{G}_1$ et $\overline{G}_2$ (voir le schéma de brochage sur la figure 7. 8).

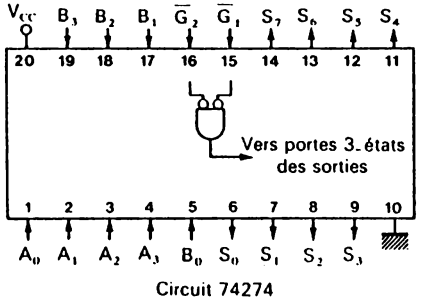


Fig. 7. 8

6. OPÉRATIONS EN BCD

Toutes les opérations que nous venons d'étudier supposent que les données et les résultats soient écrits en binaire pur.

Malheureusement ce code n'est pas pratique pour l'homme, même si l'informaticien arrive à s'accommoder de sa transcription particulière sous forme de code hexadécimal.

Le code BCD, qui transcrit les nombres décimaux en tétrades (ensembles de 4 bits) binaires est, lui, beaucoup plus compréhensible. Grâce à ce code on pourra en particulier rendre la lecture des données tout à fait claire en faisant appel à des décodeurs et des afficheurs «7 segments».

Deux possibilités s'offrent alors à nous : soit créer des circuits capables de réaliser les opérations en BCD, soit transcoder le BCD en binaire pur et vice-versa pour concilier les exigences de l'homme avec celles des circuits électroniques.

6.1 Circuits de traitement arithmétique en BCD

6.1.1 L'addition en BCD

Rappelons que le BCD n'utilise aucune des 6 tétrades qui vont de 1010_2 à 1111_2.

C'est-à-dire que dès qu'apparaîtra 1010 dans le résultat d'une addition, il faudra le transformer en 1-0000 dont la traduction en binaire pur serait 16.

D'une façon plus complète

(10)	1010	devra être traduit par	1-0000	(16)
(11)	1011	devra être traduit par	1-0001	(17)
(12)	1100	devra être traduit par	1-0010	(18)
(13)	1101	devra être traduit par	1-0011	(19)
(14)	1110	devra être traduit par	1-0100	(20)
(15)	1111	devra être traduit par	1-0101	(21)

Chaque fois que le résultat de l'addition de deux chiffres BCD dépassera 1001, il faudra donc «truquer» ce résultat en y ajoutant 6.

De plus, chaque fois que cette addition créera une retenue (résultat entre 16 et 19), il faudra procéder de même.

En effet 16 s'écrit 10000 en binaire pur, alors qu'en BCD il doit être codé 1-0110.

En définitive on devra rajouter 110_2 (6_{10}) au résultat si celui-ci est supérieur à 1001 ou s'il y a une retenue d'ordre supérieur. Dans ces deux cas, une retenue doit être transmise dans le cinquième registre du résultat.

Cette logique peut se traduire dans un circuit par le schéma de la figure 7. 9.

L'addition suivant les «normes» du binaire pur donnant un premier résultat $P_3P_2P_1P_0$, les valeurs supérieures à 1001 seront détectées par

$$(P_1 + P_2)P_3 = 1$$

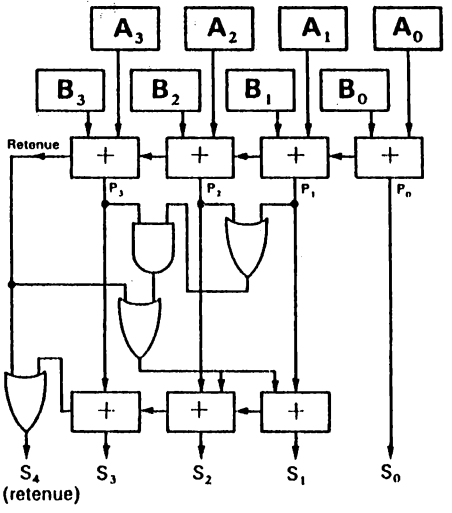


Fig. 7. 9

Ces états ou ceux entraînant une retenue d'ordre supérieur entraînent l'addition supplémentaire de $1\,1\,0_2$ et une retenue S_4.

Un tel additionneur comporte environ deux fois plus de portes que l'additionneur binaire. Il sera donc, en même temps deux fois plus lent.

On pourra essayer de construire le schéma d'un soustracteur BCD à partir des mêmes principes et par la méthode de l'addition des compléments à 1 des bits du nombre à soustraire pour constater qu'il est plus complexe encore et qu'il sera difficile de concevoir un additionneur BCD capable de faire les deux types d'opérations.

Pour ces raisons, on préfère le plus souvent disposer d'organes de calcul travaillant en binaire pur et organiser un transcodage BCD → binaire en entrée et un transcodage binaire-BCD en sortie.

6.2 Transcodage BCD → binaire

6.2.1 Transcodage par circuits logiques

On pourrait penser à définir les lois de transformation qui permettent de passer d'un nombre écrit en BCD à un nombre écrit en binaire, puis à réaliser le circuit logique qui répond à ces lois.

Prenons quelques exemples :

Supposons que le nombre BCD à transcoder soit composé de deux tétrades (4 bits) correspondant à deux nombres décimaux. Imaginons alors que nous voulons transcoder les nombres 07, 17, 27, 37, 47 etc.

Le tableau ci-dessous nous donne le résultat en BCD et en binaire pur

Décimal	BCD		Binaire pur
07	0000	0111	0111
17	0001	0111	10001
27	0010	0111	11011
37	0011	0111	100101
47	0100	0111	101111
57	0101	0111	111001
67	0110	0111	1000011
77	0111	0111	1001101
87	1000	0111	1010111
97	1001	0111	1100001

Ce tableau nous laisse présager la complexité du décodage.

En fait, si l'on prend l'un de ces nombres, 47 par exemple, il faudra ajouter aux unités ($7_{10} = 1\,1\,1_2$) le nombre équivalent à une dizaine ($1\,0\,1\,0_2$) multiplié par le nombre des dizaines.

Il faudra donc un multiplieur de 4 bits (1 0 1 0) par 4 bits (de 0 à 9_{10}), puis un additionneur.

L'ensemble risque d'être non seulement couteux, mais relativement lent.

6.2.2 Transcodage par mémoire morte

Rappelons-nous alors que nous pouvons faire appel à une autre forme de transcodage (cf. les transcodeurs BCD, 7 segments), sous forme de mémoire morte.

Cette solution a été choisie par les constructeurs de circuits intégrés, dans le circuit 74184.

Ce circuit permet de transcoder un nombre BCD de 6 bits, donc au plus le nombre 1 1-1 0 0 1 c'est-à-dire 39. Comme le bit de poids le plus faible est toujours identique en BCD et en binaire, il n'y a que 5 entrées sur le circuit, et le montage sera celui de la figure 7. 10.

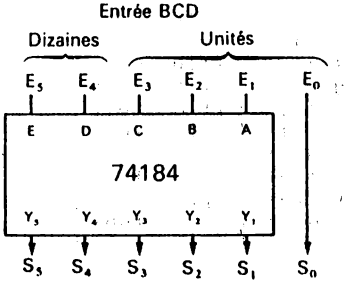


Fig. 7. 10

Si l'on veut aller plus loin dans le transcodage on peut alors associer plusieurs circuits entre eux. C'est ainsi que les documentations des fournisseurs indiquent comment réaliser un transcodeur BCD binaire ayant 6 tétrades d'entrée (de 0 à 999 999) à l'aide de 28 circuits 74184 (*fig.* 7. 11).

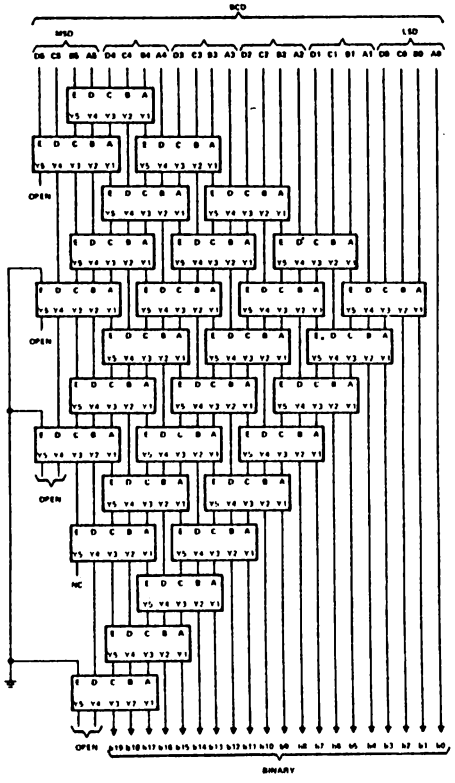


Fig. 7. 11 (Doc. Texas Instruments)

Le temps de réponse d'un tel assemblage passera alors de 70 ns (pour un seul circuit) à 400 ou 500 ns, ce qui est encore relativement raisonnable. Les inconvénients majeurs paraissent être alors l'encombrement et le coût.

6.2.3 Transcodage par compteurs

On peut enfin imaginer un transcodage utilisant en même temps un compteur BCD et un compteur binaire et opérant en trois phases :

1re phase. Le nombre BCD est chargé dans un registre d'entrée.

2e phase. Un multivibrateur est déclenché et ses tops sont comptés à la fois par un compteur binaire et par un compteur BCD.

3e phase. Lorsque le compteur BCD parvient à la valeur BCD enregistrée en entrée un comparateur décèle la coïncidence et arrête le comptage. Le contenu du compteur binaire est transféré en sortie.

Le schéma d'un tel transcodeur est donné en figure 7. 12.

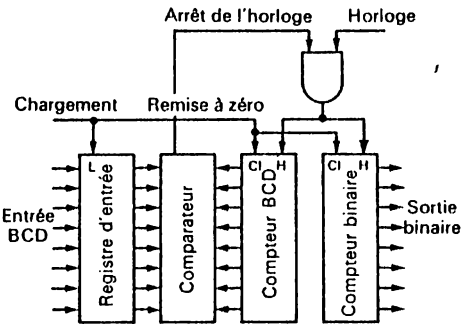


Fig. 7. 12

Si l'horloge qui incrémente les compteurs a une fréquence d'un mégahertz, on voit qu'un tel circuit pourra transcoder des nombres relativement peu élevés. Dès que l'on atteindra des nombres de l'ordre de 100 000, le délai de réponse sera déjà de 0,1 seconde, ce qui est très long à l'échelle des opérations électroniques de traitement de l'information.

Avec les nouveaux composants, notamment les circuits LSA, on peut espérer parvenir à de meilleures performances (horloges de 50 MHz par exemple) sans toutefois approcher les performances des circuits 74184.

6.3 Le transcodage binaire-BCD

Pour les mêmes raisons qu'au paragraphe précédent, nous écarterons la solution qui passerait par l'établissement d'un réseau logique de transcodage.

On pourra retenir la solution du comptage. Le schéma sera déduit de celui de la figure 7. 11 en intervertissant compteur BCD et compteur binaire puisque le nombre binaire est affiché en entrée et que son équivalent BCD doit être fourni en sortie.

Mais c'est surtout la solution de la mémoire morte qui prévaudra encore, grâce à un circuit jumeau du 184, le circuit 74185 (*fig.* 7. 13).

Celui-ci peut recevoir un nombre binaire sur 5 bits grâce à 5 entrées : A pour E_1, B pour E_2, C pour E_3, D pour E_4 et E pour E_5, l'entrée E_0 étant

directement transmise à la sortie (identité des bits unité dans le transcodage). La sortie s'opère sur 7 fils, dont le fil commun, ce qui permet de transcoder en fait de 0 à 63 (64 combinaisons binaires).

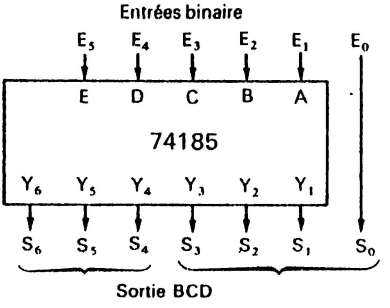


Fig. 7. 13

Là encore on peut combiner plusieurs circuits 185 pour obtenir des transcodeurs de nombres plus grands. Par exemple un système à 20 bits d'entrée (de 0 à 1048575) nécessite un réseau de 27 circuits 74185 donnant le résultat en 300 à 400 ns.

6.4 Transcodage séquentiel

Il reste une dernière méthode de transcodage qui est directement inspirée de la méthode qui nous a permis de traduire un nombre décimal en binaire pur (chapitre 2, § 6.2.2).

Prenons alors le nombre binaire pur 1 0 1 1 0 en examinant les bits successifs du plus fort au plus faible :

- le premier bit (1) nous fournit le chiffre $\boxed{1}$
- le second (0) déclenche l'opération

$$1 \times 2 + 0 = \boxed{2}$$

- le troisième (1) déclenche

$$2 \times 2 + 1 = \boxed{5}$$

- le quatrième (1) déclenche

$$5 \times 2 + 1 = \boxed{11}$$

- le cinquième (0) déclenche

$$11 \times 2 + 0 = \boxed{22}$$

le nombre binaire 1 0 1 1 0 correspond donc à 22, c'est-à-dire à 1 0-0 0 1 0 en BCD.

A partir de cette méthode on peut construire le transcodeur représenté sur la figure 7. 14. En voici son fonctionnement :

a) Le nombre à transcoder est introduit dans le registre E d'entrée, poids faible à droite, poids fort à gauche. Ce registre est en même temps registre à décalage vers la gauche. On voit que le bit de poids le plus fort sert en même temps de retenue de rang 0 à un additionneur BCD du type de celui de la figure 7. 9.

b) Par une impulsion appliquée sur R (commande Load) on charge R du résultat de l'addition obtenue. Jusqu'à présent ce résultat ne peut être que 0 ou 1.

c) Le registre est décalé d'une position vers la gauche. C'est maintenant C_3 qui sert de retenue R_0 à l'additionneur.

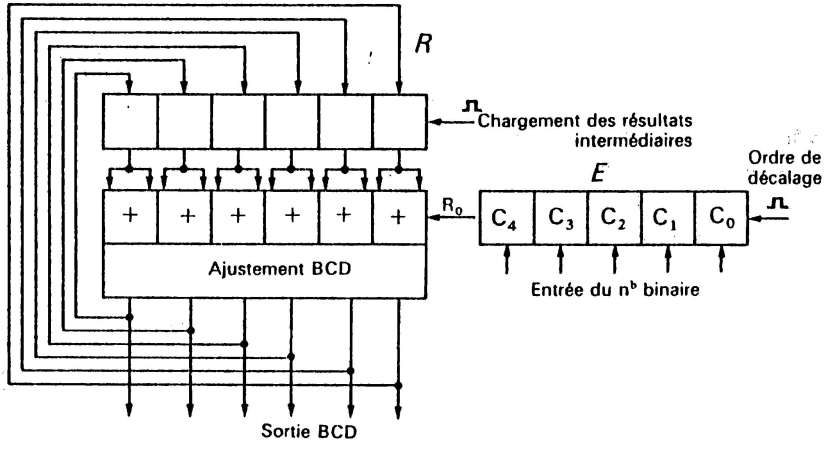


Fig. 7. 14

d) En même temps le contenu de R est transmis aux entrées de l'additionneur BCD. Le résultat est alors $2C_4 + C_3$ puisque chaque cellule de R figure sur chacune des deux entrées de chaque cellule de l'additionneur.

e) On continue le cycle *b*, *c*, *d* précédent jusqu'à ce que le nombre dans E ait complètement défilé, obtenant donc successivement

$$2(2C_4 + C_3) + C_2$$
$$2(4C_4 + 2C_3 + C_2) + C_1$$

et enfin $2(8C_4 + 4C_3 + 2C_2) + C_0$ en BCD.

La difficulté résidera dans la bonne synchronisation des opérations de chargement des registres et de décalage et, surtout, dans l'arrêt de l'opération après que C_0 soit apparu sur R_0.

EXERCICES

1. Effectuer l'addition binaire

$$\begin{array}{r} 111111 \\ +\,101111 \end{array}$$

2. Effectuer l'addition en BCD de 27 + 39.

3. Réaliser la conversion, du BCD en binaire, des nombres suivants :

a) 1111-1111,
b) 1001-1001.

4. Réaliser la conversion, du binaire en BCD, des nombres suivants :

a) 11111,
b) 1011001.

5. Effectuer la multiplication binaire suivante :

$$111111 \times 111111$$

6. Réalisation d'une mémoire de transcodage binaire-BCD

Nous avons vu (§ 6.2.3) comment on pourrait transcoder du binaire en BCD ou du BCD en binaire grâce à des compteurs.

Nous avons également vu comment on pouvait effectuer ce transcodage grâce à une mémoire morte.

L'idée de cet exercice est d'utiliser les deux techniques en complément l'une de l'autre de la façon suivante :

a) *Remplissage de la mémoire*

On utilise, pour cela, le montage à deux compteurs synchronisés. Le compteur binaire est relié aux adresses de la mémoire et le compteur BCD est relié aux entrées, en écriture, des mots mémoire.

b) Lorsque toute la mémoire a été remplie, on relie adresses et sorties mémoires aux circuits utilisateurs. On a ainsi obtenu un circuit de transcodage à mémoire.

Imaginer le circuit complet en utilisant les mémoires TMS 4043 (256 mots de 4 bits) et les compteurs 74160 ou 162 (BCD) et 74161 ou 163 (binaires).

CHAPITRE 8

La programmation des opérations de traitement

INTRODUCTION

Une machine électronique qui ne serait capable que d'additionner 2 chiffres décimaux, de les soustraire ou de les multiplier ne serait d'aucun intérêt pour l'homme.

Il faut étendre la capacité de la machine au traitement de nombres importants.

Si l'on veut additionner deux nombres de 10 chiffres à travers un additionneur, il faudra deux registres d'entrée à 40 éléments binaires (10 tétrades pour coder en BCD), ainsi qu'un additionneur à 40 étages d'addition et une sortie sur 41 fils. L'augmentation de capacité se traduirait alors par un encombrement et un coût prohibitifs.

Nous allons voir, autour de l'exemple de l'addition, comment augmenter presque indéfiniment cette capacité tout en ne disposant que d'un simple additionneur BCD à 4 étages mais en utilisant les possibilités offertes par la logique séquentielle. Il s'agit en fait d'organiser, à la suite les unes des autres dans le temps, les opérations élémentaires qui conduisent au résultat final. On dira aussi que l'on «programme» une suite d'opérations.

1. ANALYSE DES OPÉRATIONS ÉLÉMENTAIRES REQUISES POUR UNE ADDITION

Faisons tout d'abord cette analyse sans nous préoccuper de la machine, mais en essayant de lister les opérations que réalise l'homme pour obtenir le résultat d'une opération, dans l'ordre où il la réalise.

1.1 Processus humain de réalisation d'une addition de deux nombres

a) Sauf cas très particulier (prodige) l'homme va tout d'abord inscrire les deux nombres à additionner. Cette inscription joue le rôle de mémoire.

b) L'homme additionne d'abord les deux chiffres des unités et reporte le résultat, au-dessous de ces chiffres, en portant en mémoire (soit mentalement, soit par écrit) la retenue éventuelle.

c) Il additionne ensuite les 2 chiffres des dizaines et y rajoute la retenue précédente. Il reporte de la même façon le résultat et la retenue.

d) Il va ainsi des rangs de poids faibles aux rangs de poids forts jusqu'à ce qu'il n'y ait plus de chiffre à additionner. Il inscrit alors la dernière retenue.

e) Il lit alors le résultat sur la feuille de papier qui, là encore, joue le rôle de mémoire.

Dans tout ce processus, l'homme n'a jamais fait autre chose que d'additionner deux chiffres décimaux entre eux, puis d'ajouter ou non une retenue de 1 au résultat. Il ne s'est servi, en fait, que d'un seul additionneur, celui qu'il a acquis comme un automatisme sur les bancs de l'école ou chez lui en répétant sa «table d'addition».

1.2 Transposition du processus humain à la machine

a) Inscription des nombres à additionner

La machine étant au service de l'homme, ce dernier ne doit pas avoir plus de difficulté à introduire les nombres en machine qu'il n'en avait à les inscrire sur une feuille de papier. Il n'est donc pas question de l'obliger à un transcodage en binaire ou même en BCD.

Cette opération se fera presque toujours en utilisant les touches d'un clavier où chaque chiffre à transmettre est inscrit en clair.

L'étude de cet enregistrement par clavier sera l'objet du chapitre 9. Nous supposerons donc ici le problème résolu : les deux nombres figurent alors en machine, chaque chiffre ayant été traduit en BCD sous forme de 4 bits.

b) Addition des deux chiffres des unités

La forme même de la mise en mémoire (4 bits BCD pour chaque chiffre) nous oriente vers l'utilisation d'un additionneur BCD tel qu'il a été décrit au chapitre précédent.

Le résultat de l'addition sera fourni par le registre mémoire à 4 bits de sortie et le fil de retenue.

c) Addition des chiffres de rang supérieur

Le même additionneur devant servir à nouveau pour ces additions il faut mettre en mémoire, autre part que dans le registre de sortie, la valeur obtenue lors de l'addition précédente et reporter en entrée de l'addition suivante la retenue éventuelle.

La nouvelle addition peut alors être effectuée avec la tétrade correspondant aux chiffres des dizaines.

Ce même processus doit ensuite se dérouler pour chaque couple de chiffres de rang supérieur jusqu'au dernier.

d) Résultat

Enfin le résultat doit pouvoir être lu en clair par l'homme. On pourra utiliser ici le procédé classique de l'affichage « sept segments » avec ou sans multiplexage.

2. RÉALISATION PRATIQUE D'UN ADDITIONNEUR DE DEUX NOMBRES DE HUIT CHIFFRES

2.1 Introduction des nombres en mémoire

Nous supposons résolue la frappe des nombres sur un clavier et leur introduction dans la machine.

Il faut que ces nombres soient rangés dans une mémoire.

Chaque nombre ayant 8 chiffres, il sera codé sous forme de 8 tétrades (8 × 4 bits). Si nous adoptons une mémoire organisée en mots de 4 bits, il nous faudra au moins 16 mots, ce qui ne laisse aucune place pour d'autres valeurs telles que les résultats.

Nous choisirons donc une mémoire de plus grande capacité, c'est-à-dire celle qui dispose de 256 mots de 4 bits.

Plusieurs modèles sont à notre disposition : TTL ou MOS, différant par leur temps d'accès. Pour des raisons de commodité nous emploierons, par exemple, une MOS statique à alimentation unique de 5 V, compatible TTL dont de nombreux modèles sont proposés par les divers fabricants.

Sachant à l'avance que nous aurons à additionner successivement les 4 bits des « unités » de chaque nombre, puis les 4 bits des dizaines, etc. nous allons introduire ces valeurs en mémoire de la façon suivante :

adresse 00000 : les 4 bits unités de A
adresse 00001 : les 4 bits unités de B
adresse 00010 : libre
adresse 00011 : libre
adresse 00100 : les 4 bits dizaines de A
adresse 00101 : les 4 bits dizaines de B
adresse 00110 : libre
adresse 00111 : libre
adresse 01000 : les 4 bits centaines de A
adresse 01001 : les 4 bits centaines de B
etc. jusqu'à l'adresse 11111 (31)

2.2 Le circuit additionneur

Ce circuit comportera l'additionneur binaire à 4 étages et le circuit de transcodage du résultat en BCD. Nous en avons vu son principe au chapitre précédent (*fig.* 7. 9).

Il peut être réalisé avec 2 circuits 7483 dits « full adders » de 4 bits suivant le schéma de la figure 8. 1.

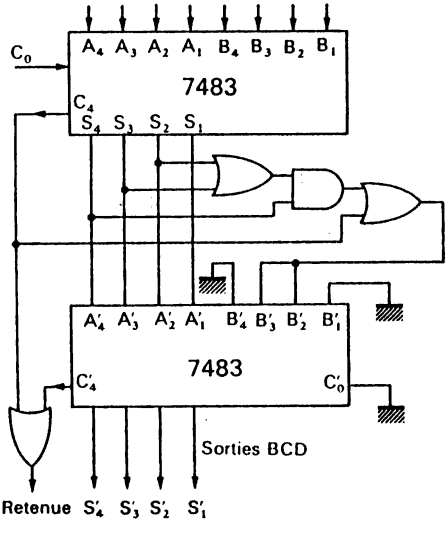


Fig. 8. 1

On fera attention au brochage de ces circuits (*fig.* 8. 2) et notamment à l'emplacement des bornes d'alimentation (borne 5 pour V_{cc} et 12 pour la masse).

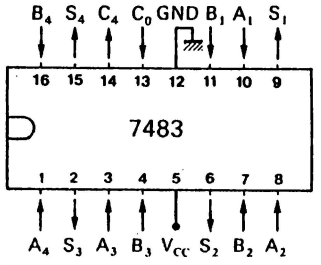


Fig. 8. 2

2.3 Liaison entre les données en mémoire et l'additionneur

Pour obtenir le résultat de l'addition de deux nombres de 4 bits, A et B, dans l'additionneur, il faut que ces deux nombres soient présents simultanément aux entrées A_1 à A_4 et B_1 à B_4 de cet additionneur.

Or nous ne pourrons lire les valeurs en mémoire que l'une après l'autre.

Il faut donc au moins un registre de stockage intermédiaire entre la mémoire et l'additionneur, registre dans lequel sera mémorisée la première donnée (A), jusqu'à ce que B apparaisse.

Nous opterons même pour deux registres, un pour A et un pour B, ce qui nous laissera plus de temps pour exploiter le résultat apparaissant en sortie de l'additionneur.

Ces deux registres (*fig.* 8. 3) seront raccordés en entrée au bus des données en provenance de la mémoire.

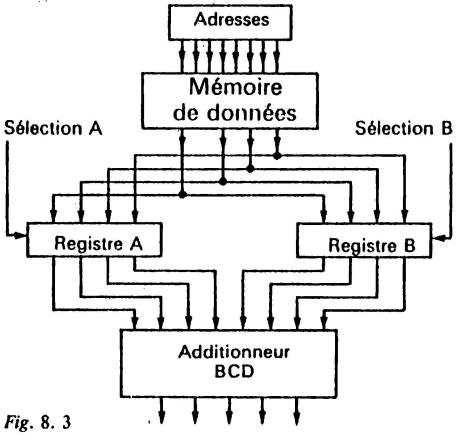


Fig. 8. 3

Il faudra sélectionner ou inhiber l'entrée des données sur chacun des registres en fonction de l'adresse envoyée sur le décodeur d'adresse de la mémoire.

Si l'adresse est 0, 100, 1000, 1100, etc. (se terminant par deux 0), la donnée sélectionnée est un chiffre de A, c'est le registre A qui doit être sélectionné.

Si par contre l'adresse se termine par 01 (01, 101, 1001, 1101 etc.) c'est le registre B qui doit être sélectionné.

2.4 Prise en compte de la retenue

La retenue apparaissant sur le cinquième fil des résultats de l'addition doit être prise en compte et réinjectée, lors de l'addition suivante, en C_0.

Il faudra donc un dispositif de mise en mémoire de cette retenue.

2.5 Synchronisation des additions successives

L'ensemble du système sera sous la dépendance d'une horloge qui va rythmer les différentes phases de l'opération. Mais le bon déroulement de toutes les opérations dépendra de la mise au point d'un ensemble de portes logiques assez complexe.

On peut bâtir cet ensemble sur le principe qui a présidé à la conception du système de synchronisation de l'exercice 6 du chapitre précédent (réalisation d'une mémoire de transcodage binaire-BCD).

Par exemple, ayant mis un compteur (type 161 ou 163) à 0, la première adresse de ce compteur (00000) est envoyée sur le sélecteur d'adresse de la mémoire.

Environ 500 ns après cet adressage, on pourra charger le mot de mémoire dans le registre A, et ceci, 50 à 100 ns avant que l'adresse ne change. Après quoi on passera à l'adressage et à la lecture de l'adresse 00001. A chaque fois des signaux d'autorisation, de chargement ou d'inhibition devront être envoyés, signaux qui devront souvent être inversés, composés avec d'autres ou retardés à travers des monostables.

Ce type de solution est toujours possible, mais il n'est jamais simple. De plus l'électronicien répugne, en matière de synchronisme, à utiliser des retardateurs (type 121) à base de cellules RC pour la raison que ces composants sont quelquefois encombrants, mais surtout parce qu'ils vieillissent mal et peuvent donc dérégler la belle construction.

Aussi le type de solution que nous allons maintenant vous présenter est-il devenu le plus employé.

2.5.1 *Liste des micro-opérations*

Le premier travail consiste à dresser de façon précise la liste de toutes les opérations élémentaires nécessaires pour parvenir au résultat escompté.

Par opération élémentaire il faut entendre l'autorisation donnée à la sortie d'informations sur un bus d'adresses ou de données vers un autre registre, l'incrémentation d'un compteur par **l'envoi d'une impulsion sur son horloge, une mise** à 0 d'une bascule, etc.

Il s'agit ici de décrire successivement 8 cycles de lecture des blocs de 4 bits de A et de B, d'addition de ces blocs, d'inscription du résultat et de report de la retenue.

Au départ on supposera tous les compteurs et bascules mis à 0. L'horloge va toujours rythmer la succession des opérations à travers un compteur binaire type 74161. Nous appellerons ce compteur « compteur d'opérations ».

La *première opération* va consister à envoyer une adresse 00000 sur la mémoire de données.

Cette adresse ne peut pas être prélevée à la sortie de notre compteur d'opérations, car nous voulons pouvoir incrémenter l'adresse à un moment ou à un autre, mais non à chaque coup d'horloge. Nous devons donc disposer d'un *compteur d'adresses* distinct du compteur d'opérations.

La *deuxième opération* consistera à lire les données figurant dans la mémoire de données à l'adresse 00000 (unités de A) et à les entrer dans le registre A.

La *troisième opération* consistera à envoyer l'adresse 00001 sur la mémoire. On la réalise en envoyant une impulsion sur l'horloge du compteur d'adresses.

La *quatrième opération* chargera le registre B de la valeur lue en mémoire.

Notons que A et B étant chargés, le résultat de l'addition apparaît quelques 50 ns après, en sortie de l'additionneur, pour peu que la retenue C_0 en entrée soit correcte. Ici C_0 est nulle, ce 0 étant obtenu par la remise à 0 initiale. Il faudra songer par la suite à mettre, dans un registre mémoire, la nouvelle retenue apparaissant sur le cinquième fil de résultat (voir *fig.* 8. 1).

En *cinquième opération* il s'agira donc de lire le résultat de la première opération et de le charger dans une mémoire. On pourrait évidemment penser à effectuer tout de suite un affichage 7-segments, mais ce serait compliquer à souhait cette séquence, avec un problème supplémentaire non négligeable qui serait la lisibilité du résultat. En effet, l'ensemble de toutes les opérations va durer une fraction de seconde, temps très insuffisant pour lire un résultat quelconque. Aussi faudrait-il répéter la séquence plusieurs milliers de fois pour en permettre la lecture. Sinon cette lecture impliquera la présence de 8 registres de

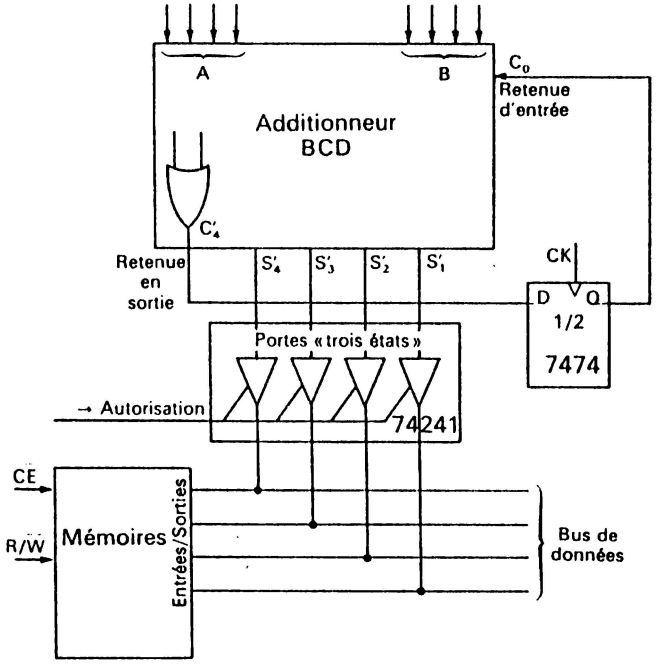


Fig. 8. 4

4 bits plus 1 registre pour la retenue finale; autant mettre alors ces résultats dans la mémoire pour les exploiter ensuite, lors d'une autre séquence.

Le résultat sera donc chargé en mémoire 0 0 0 1 0 laissée disponible lors du chargement des données.

On réajustera donc la liste des opérations de la façon suivante :

Cinquième opération : incrémenter le compteur d'adresses

Sixième opération : mettre la mémoire en écriture pour enregistrer le résultat de l'addition (songer à autoriser le résultat sur le bus de données).

En *Septième opération* il faut penser à mettre en mémoire la retenue pour pouvoir la réintroduire lors de l'addition suivante.

En imaginant que nous disposions d'une bascule D pour mise en mémoire de cette retenue, bascule dont l'entrée D est reliée en permanence au cinquième fil de l'additionneur, il faudra envoyer une impulsion sur l'horloge de cette bascule pour obtenir C_0 en sortie Q.

Le détail de ces opérations n'est pas indépendant de la configuration des circuits, aussi avons nous décrit sur la figure 8. 4 les circuits concernant le raccordement du résultat et de sa retenue aux autres organes du système (mémoire de données, entrée C_0 de retenue, bascule D de mise en mémoire de la nouvelle retenue).

Dès lors la *septième opération* est la suivante :

Septième opération : incrémenter le compteur d'adresses et envoyer une impulsion sur l'horloge de la bascule D de retenue.

A ce stade le premier cycle d'opérations paraît terminé. Pourtant, il ne faut pas oublier d'incrémenter à nouveau le compteur d'adresses pour qu'au début du second cycle ce soit bien le bloc des bits de dizaine de A qui soit adressé.

A ce propos on n'oubliera pas non plus de vérifier que l'on n'est pas parvenu à la fin du huitième cycle, auquel cas il faudra arrêter les opérations.

Comme lors de chaque cycle, on incrémente quatre fois le compteur d'adresses, il y aura eu 32 incrémentations à la fin du huitième cycle et ce compteur affichera donc la valeur 1 0 0 0 0 0. L'apparition du 1 pour la première fois, en rang 6, devra être le signal d'arrêt. D'où :

Huitième opération : incrémenter le compteur d'adresses.

Neuvième opération : si l'adresse de sortie du compteur d'adresse est 1 0 0 0 0 0, arrêter le processus (passer éventuellement au processus d'affichage des résultats). En même temps la dernière retenue éventuelle pourra être mise en mémoire dans un registre spécial (cette opération peut être automatisée par l'apparition d'un 1 sur le sixième fil du compteur d'adresses).

La liste complète des opérations sera donc la suivante :

Affichage du compteur d'opérations	Opération effectuée
(0) 0 0 0	— Le compteur d'adresses (remis à 0) affiche l'adresse des unités de A.
(0) 0 0 1	— Un ordre « Load » est envoyé sur le registre A relié au bus de données de la mémoire.
(0) 0 1 0	— Le compteur d'adresses est incrémenté.
(0) 0 1 1	— Un ordre « Load » est envoyé sur le registre B.
(0) 1 0 0	— Le compteur d'adresses est incrémenté.
(0) 1 0 1	— Le résultat de l'additionneur est autorisé sur le bus de données par des portes « 3 états » et la mémoire est mise en écriture ($\overline{W} = 0$).
(0) 1 1 0	— Le compteur d'adresse est incrémenté. Une impulsion est envoyée sur Ck du registre de retenue, appliquant cette retenue sur l'additionneur.
(0) 1 1 1	— Le compteur d'adresse est incrémenté. Si le 6e bit d'adresse passe à 1, ceci arrête les opérations et met en mémoire la dernière retenue.

2.5.2 *Caractéristiques des commandes des micro-opérations*

Ces opérations vont être effectuées grâce à des impulsions, positives ou négatives ou des basculements de signaux. Ces différentes commandes (ou « contrôles » en utilisant un néologisme anglo-saxon) dépendent des circuits utilisés, c'est pourquoi il est nécessaire de définir ces circuits :

La mémoire de 256 mots de 4 bits sera une TMS 4043. Elle est munie de deux commandes :

$\overline{CE}$ (borne 13) que nous supposerons à la masse, ce qui active en permanence la mémoire.

R/$\overline{W}$ (borne 14), autorisant la lecture au niveau 1 et l'écriture au niveau 0.

Les registres tampons A et B seront des 7495. Pour les charger en parallèle il faut que la commande « Mode Control » soit à 1 (nous la branchons au +) et que la commande « Load » (borne 8) passe de 1 à 0.

Le compteur d'adresses sera un 74161. Il est muni d'une commande « $\overline{Clear}$ » (borne 1) active à 0, d'une horloge « Ck » (borne 2) qui incrémente la sortie à chaque front montant de cette horloge, et de circuits d'autorisation « Enable P » (borne 7) et « Enable T » (borne 10) que nous relierons au + 5 V.

Notons que le compteur devant être composé de deux circuits 74161 (il doit compter 32 adresses), le circuit de poids forts verra sa borne Ck reliée à la sortie 15 (Ripple carry output) du premier.

Pour autoriser le transfert du résultat de l'addition sur le bus des données (4 bits) nous pourrons utiliser la moitié des portes d'un circuit 74241, en choisissant la commande active au niveau 1 (borne 19).

Le schéma général d'implantation des circuits avec leurs commandes est représenté sur la figure 8. 5.

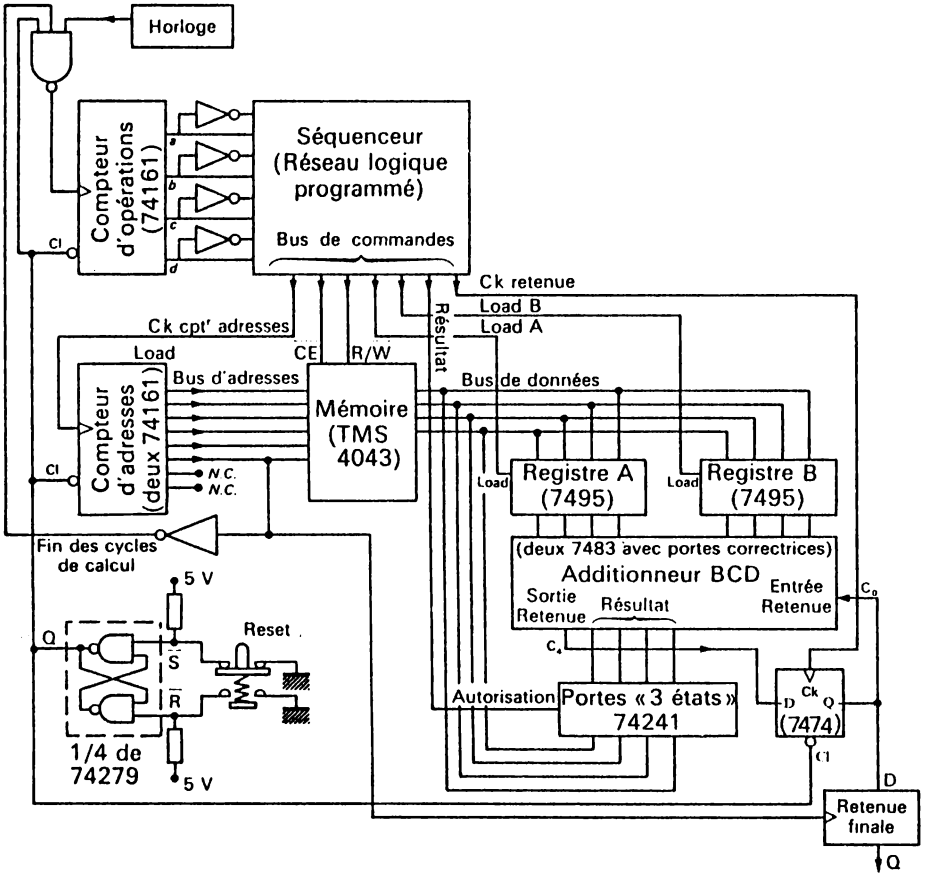


Fig. 8. 5

2.5.3 Génération des commandes

Pour générer ces commandes il faut qu'à chaque valeur affichée au compteur d'opérations, des 0 ou des 1 soient envoyés sur chacun des fils de commande en fonction des opérations à réaliser. A partir du tableau du § 2.5.1 et des caractéristiques vues au § 2.5.2, on peut dresser le nouveau tableau ci-dessous.

Affichage du compteur d'opérations	Compteur d'adresses de la mémoire	Mémoire Read = 1 Write = 0	Chargement registre A L = ↓	Chargement registre B L = ↓	Résultat sur bus données (porte 3 états à 1)	Retenue sur additionneur Ck = ⎍
N°						
0 (0)000	0	1	1	1	0	0
1 (0)001	0	1	[0]	1	0	0
2 (0)010	[1]	1	1	1	0	0
3 (0)011	0	1	1	[0]	0	0
4 (0)100	[1]	1	1	1	0	0
5 (0)101	0	[0]	1	1	[1]	0
6 (0)110	[1]	1	1	1	0	[1]
7 (0)111	[1]	1	1	1	0	0

Nous avons là ce que l'on peut appeler de termes différents, mais qui recouvrent, bien sûr, un objet unique :

— Soit la table de vérité d'un circuit logique à 3 entrées (les 3 bits de poids faible du compteur d'opérations) et à 6 sorties (les commandes).

— Soit le descriptif du contenu d'une mémoire de 8 mots de 6 bits.

— Soit un décodeur, qu'on peut appeler aussi séquenceur parce que le décodage se fait ligne après ligne. Ce décodeur ordonne dans le temps la séquence des opérations à exécuter.

2.5.4 Chronogramme des opérations et examen critique

Il est commode et prudent de représenter la séquence des opérations sous la forme d'un chronogramme. Cette représentation va nous révéler quelques défauts dans l'ordonnancement des opérations (*fig.* 8. 6).

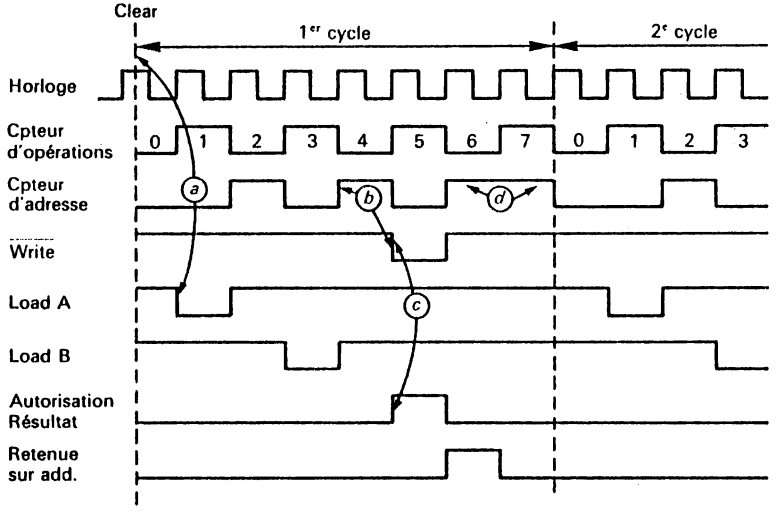


Fig. 8. 6

a) Le « Clear » général ou « Reset » est supposé réalisé à la main, à l'aide d'un interrupteur. Le démarrage peut donc survenir à n'importe quel moment du cycle de l'horloge. S'il intervient juste avant que le signal d'horloge ne passe au niveau haut, le compteur d'adresses va rester très peu de temps à 0000, et au pire le temps d'un cycle d'horloge. Les données contenues dans la mémoire à l'adresse 0000 n'auront pas eu le temps d'être « stables » en sortie si le temps d'accès dépasse cette durée de cycle. Il faut donc prendre quelques précautions.

– En synchronisant le Clear général avec l'horloge. Pour éviter le risque de rebonds de l'interrupteur nous utiliserons la précieuse bascule RS (dont 4 exemplaires sont contenus dans le circuit 74279). Le circuit de Reset est représenté en figure 8. 7.

– En retardant l'impulsion envoyée au compteur d'adresses pour laisser le temps aux données de la mémoire 0000 d'être stables sur les sorties.

b) L'écriture du résultat de l'addition se fait après une seule période d'horloge. Là encore, la mémoire ayant un temps d'accès de 650 ns, on risque de n'enregistrer que des résultats fantaisistes si la fréquence d'horloge est supérieure à 1,5 MHz. Cette fréquence peut être considérée comme très raisonnable pour notre application, mais nous verrons plus loin pourquoi on cherche à utiliser des fréquences bien plus grandes.

De toute façon il est prudent, là encore, de retarder la mise en écriture après que l'adresse ait été affichée.

c) Le résultat n'est autorisé sur le bus de données qu'au moment où la mémoire est mise en écriture. Pour peu que la porte 3-états mette un certain délai à réagir, ou que l'ordre lui parvienne avec un peu de retard, l'écriture aura lieu alors que les données à inscrire ne sont pas stables, ce qui est à éviter.

Il convient donc d'autoriser les résultats une période avant. Mais ces données vont alors se trouver en conflit avec les sorties de mémoire et provoquer des courts-circuits néfastes. Il faut donc appliquer un ordre $\overline{CE} = 1$ pour inhiber les sorties de mémoire tant que l'ordre d'écriture n'est pas donné. Ceci exigera une nouvelle commande ($\overline{CE}$) et donc un nouveau fil de commande.

d) Le défaut majeur concerne les deux tops successifs envoyés au compteur d'adresses en périodes 6 et 7. Comme ces deux tops sont jointifs, ils n'en feront qu'un et nous n'aurons qu'une seule incrémentation de compteur au lieu de deux!

Il faut toujours séparer deux commandes successives par un retour à l'état de repos.

2.5.5 Établissement d'une séquence corrigée

Compte tenu des remarques précédentes, il n'est plus possible de se restreindre à 8 périodes d'horloge. La logique binaire nous oblige alors à passer à 16, quitte à glisser des temps d'attente où nulle opération n'est réalisée (commande « No-opération »).

La nouvelle séquence sera alors la suivante :

Ce séquenceur peut être maintenant commandé par une horloge de 2,5 MHz de fréquence. Pour effectuer les 8 cycles d'addition il faudra, dans ce cas, environ 5 millisecondes!

2.6 Réalisation du séquenceur

Il serait possible d'utiliser, pour réaliser le séquenceur, une mémoire de 16 mots de 7 bits (16 × 8 en fait), c'est-à-dire un groupe de deux mémoires de 16 mots de 4 bits, disposant de son

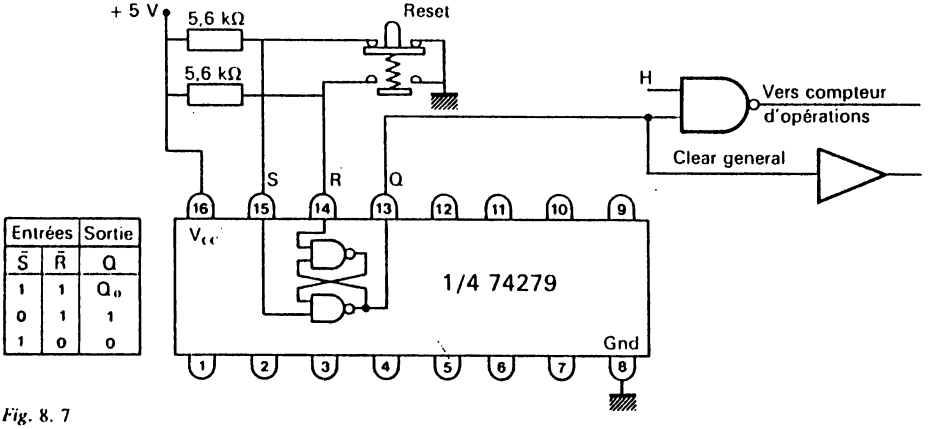


Fig. 8. 7

N° d'opération	Affichage du compteur d'opérations	Ck compteur d'adresses	$\overline{CE}$ mémoire	R/$\overline{W}$ mémoire	« Load » Registre A	« Load » Registre B	Autorisation résultat sur bus données	Ck retenue sur additionneur
RESET GÉNÉRAL								
0	0000	0	1	1	1	1	0	0
1	0001	0	1	1	1	1	0	0
2	0010	0	0	1	1	1	0	0
3	0011	0	0	1	1	1	0	0
4	0100	0	0	1	0	1	0	0
5	0101	1	0	1	1	1	0	0
6	0110	0	0	1	1	1	0	0
7	0111	0	0	1	1	0	0	0
8	1000	1	1	1	1	1	0	0
9	1001	0	1	1	1	1	1	0
10	1010	0	0	0	1	1	1	0
11	1011	0	0	1	1	1	0	0
12	1100	1	1	1	1	1	0	0
13	1101	0	1	1	1	1	0	1
14	1110	1	1	1	1	1	0	0
15	1111	0	1	1	1	1	0	0

sélecteur d'adresses et de ses sorties. Mais il faudrait aussi, dans ce cas, disposer du matériel nécessaire pour réaliser cette mémoire à partir d'une PROM.

Une autre solution consiste à construire un réseau de portes logiques donnant les valeurs des commandes à partir des 4 bits de sortie du compteur d'opérations.

Nous allons étudier deux variantes de cette solution.

2.6.1 *Traduction directe du comptage en commande*

Appelons *abc* et *d* les 4 bits délivrés par le compteur d'opérations, *a* ayant le poids le plus faible.

Les seize époques de comptage de 0 à 15 correspondent aux 16 cases d'un tableau de Karnaugh à 4 entrées, suivant la disposition de la figure 8. 8.

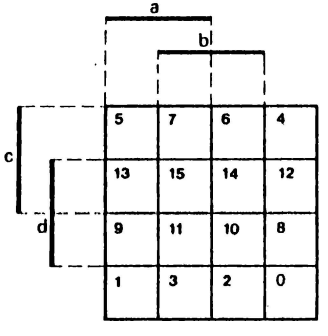


Fig. 8. 8

En se reportant au tableau de séquence (§ 2.5.5) on voit que l'horloge du compteur d'adresses doit recevoir une impulsion aux époques 5, 8, 12 et 14, ce qui se traduit par les cases en grisé de la figure 8. 9.

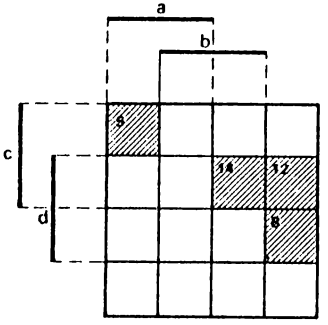


Fig. 8. 9

En appelant C_0 cette commande, on doit donc avoir

$$C_0 = \bar{a}bc\bar{d} + \bar{a}cd + \bar{a}\bar{b}d$$

Cette relation peut être directement obtenue par une série de portes « Et » et « Ou » à partir de a, b, c, d et $\bar{a}, \bar{b}, \bar{c}, \bar{d}$ (cf. *fig.* 8. 10).

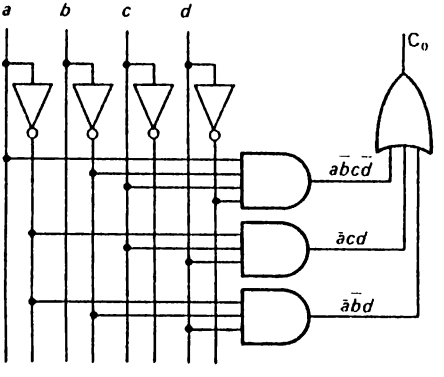


Fig. 8. 10

On procèdera de même pour les six autres commandes. L'ensemble de ce réseau logique délivrera les commandes au rythme des impulsions du compteur d'opérations; on l'appelle un « réseau logique programmé » (PLA en anglais).

Quant à la série des mots de 7 bits présentés sur le tableau du § 2.5.5, elle constitue ce que l'on appelle le **programme** du cycle d'opérations nécessaire à la réalisation d'une addition.

2.6.2 Démultiplexage à partir du compteur

La solution précédente est grande consommatrice de portes et de fils de liaison.

On peut limiter notablement les circuits en profitant de ceux que nous proposent les fabricants.

La solution consiste à utiliser les 4 bits du compteur d'opérations pour commander un démultiplexeur « une sortie parmi 16 » du type 74154 (*fig.* 8. 11). En mettant à la masse les bornes G_1 et G_2, les sorties 0 à 15, dont le niveau normal est haut, passent successivement à 0 pendant la période correspondant à leur numéro.

Il est facile de créer les commandes à partir de ces seize sorties.

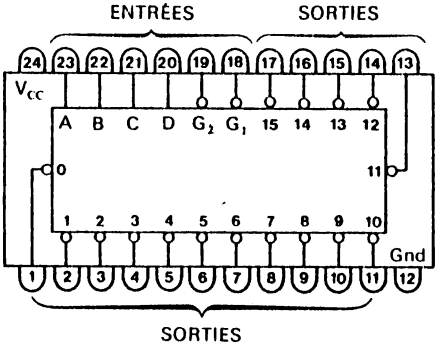


Fig. 8. 11

Par exemple la commande C_0 devant être haute aux époques 5, 8, 12 et 14 sera obtenue à partir des sorties S_5, S_8, S_{12} et S_{14} et correspond à

$$C_0 = \overline{S_5} + \overline{S_8} + \overline{S_{12}} + \overline{S_{14}} = \overline{S_5 \cdot S_8 \cdot S_{12} \cdot S_{14}}$$

La moitié d'un circuit 7420 suffira pour obtenir cette commande.

Certaines autres commandes, comme « $R/\overline{W}$ », « Load A » ou « Load B » pourront être directement branchées sur les sorties correspondantes du 74154.

On obtient alors le circuit logique programmé de la figure 8. 12.

2.7 Bilan de la réalisation de l'additionneur de nombres de huit chiffres

Nous venons d'étudier la réalisation d'un microprocesseur spécialisé et donc limité dans ses possibilités.

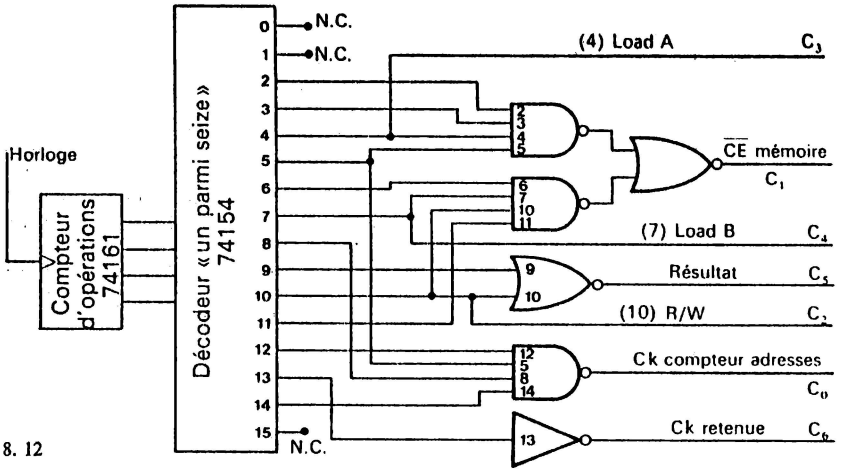


Fig. 8. 12

Tel qu'il est (*fig.* 8. 5) il présente déjà bon nombre des organes d'un microprocesseur beaucoup plus polyvalent. Il dispose :

a) d'une horloge interne qui rythme toutes ses opérations,

b) d'un compteur qui ordonne entre elles toutes les petites opérations successives (micro-opérations). On appelle ce compteur un *compteur ordinal*,

c) d'une mémoire d'opérations ou d'un réseau logique programmé qui traduit chaque donnée du compteur ordinal en commandes à travers ce que l'on appelle déjà un programme ou plutôt un *micro-programme*,

d) d'un bus de commandes (ou bus de contrôle), ici constitué des fils C_0 à C_6 qui véhiculent les ordres envoyés par le micro-programme,

e) des organes activés par ces commandes, parmi lesquels nous trouvons :

— un compteur d'adresses avec son bus d'adresses,

— une mémoire reliée à ce bus d'adresses, avec son bus de données,

— des registres-tampons pour les données,

— une unité de calcul (ici 1 additionneur BCD),

— un registre « verrouillant » la retenue finale de l'addition,

— des portes trois états évitant les courts-circuits d'informations,

— un circuit d'interruption de fin du cycle de calcul,

f) d'une commande « Reset » permettant d'initialiser l'opération (commande manuelle).

Organisée ainsi, cette petite machine peut paraître déjà compliquée, mais il faudrait beaucoup plus de circuits si nous voulions construire un additionneur BCD de deux nombres de 32 bits et obtenir le résultat par une seule opération de commande.

Nous venons d'effectuer un premier pas dans le raisonnement qui a certainement guidé vers la réalisation des microprocesseurs polyvalents qui ont fait exploser l'informatique.

Ce raisonnement est le suivant : faire des traitements complexes avec des organes simples et remplacer ainsi les organes complexes de traitement par des successions, même longues d'opérations élémentaires, gérées par un programme.

Si, de plus, on réussit à déclencher chaque opération et à la réaliser en un temps très court, au rythme d'une horloge fonctionnant à plusieurs dizaines de mégahertz, on gagnera sur l'architecture d'ensemble sans trop perdre sur les délais d'obtention des résultats.

Mais l'additionneur à 8 chiffres obtenu présente encore beaucoup d'inconvénients :

— Il exige que les deux nombres à additionner soient rangés en mémoire interne dans un ordre pré-établi.

— Cette opération de rangement (que nous n'avons pas étudiée) oblige à des agencements supplémentaires comme l'isolement du bus d'adresse de la sortie du compteur d'adresse, et son rattachement à un registre spécial.

— Il est beaucoup trop spécialisé pour présenter un intérêt quelconque. Bien sûr il peut additionner des nombres jusqu'à 8 chiffres, mais il ne peut accepter de nombres à 9 chiffres et au-delà.

2.8 Vers une plus grande polyvalence des traitements

Pour faciliter la mise en mémoire, il faut que la mémoire elle-même soit extérieure à l'unité de traitement et que l'on puisse y accéder directement pour y placer les données aux adresses qu'il nous plaira de choisir.

Pour rendre l'unité de traitement plus polyvalente, pourquoi ne pas la doter de l'unité arithmétique et logique (UAL ou ALU en anglais) du type 74181 plutôt que d'un simple additionneur, fut-il BCD. D'autres organes s'avèreront en outre utiles, comme des registres à décalage dont nous avons vu le rôle dans la multiplication.

En revanche nous limiterons considérablement nos ambitions quant aux opérations (cycles de micro-opérations) à faire exécuter, tour à tour, par l'unité de traitement.

Ainsi, avant même de déclencher une addition, nous demanderons que soient exécutées les opérations suivantes :

- Mettre la retenue d'entrée de l'UAL à 0.
- Transférer le chiffre unité de A dans le premier registre de l'UAL.
- Transférer le chiffre unité de B dans le second registre de l'UAL.
- Transférer le résultat de l'addition à une adresse donnée de la mémoire.
- Reporter la retenue sur l'entrée de l'UAL etc.

Pour chacune de ces opérations, l'unité de traitement devra posséder son cycle de micro-opérations, soit sous forme d'une mémoire ROM interne, soit sous forme d'un réseau logique programmé, soit encore sous une forme combinant les deux.

Nous développerons plus loin quelques aspects de ces solutions.

Le premier problème à résoudre est maintenant l'entrée en mémoire, de façon commode pour l'homme, des données à traiter ou des ordres à donner à l'unité de traitement. C'est ce que nous abordons dans le chapitre 9.

CHAPITRE 9

Entrée des données : le clavier

INTRODUCTION

Pour transmettre manuellement des données à un système électronique de traitement, le moyen le plus simple est d'utiliser un clavier. Encore faut-il que l'utilisation de ce clavier soit la plus évidente pour l'homme : touches codées en clair (alors que l'information transmise sera en binaire), mise en mémoire quasi-automatique, etc.

Nous allons voir comment résoudre la plupart des problèmes liés à cette technique de transmission de données.

1. CODAGE DIRECT DES TOUCHES

Imaginons que nous ayons besoin d'un clavier analogue à celui des petites calculatrices, comportant 10 touches pour les chiffres de 0 à 9 et les touches de signes (+, −, ×, ÷, =) plus une touche pour la virgule. Nous aurons donc 16 touches.

Chaque touche pourra générer un code à 4 bits, en réservant les codes binaires de 0000 à 1001 pour les chiffres et les suivants pour les signes.

Pour générer directement ce code par la touche, dans l'hypothèse où nous n'avons que 4 fils de sortie, nous pouvons nous servir d'un modèle désormais classique, celui de la ROM à diodes (*fig.* 9. 1).

Constatons tout de suite que si aucune touche n'est enfoncée la sortie génère en permanence le code 1111 que nous avons affecté à un signe. Il faut donc une sortie supplémentaire qui n'autorise la sortie du résultat (et pourra aussi déclencher sa mise en mémoire) que si une touche est enfoncée.

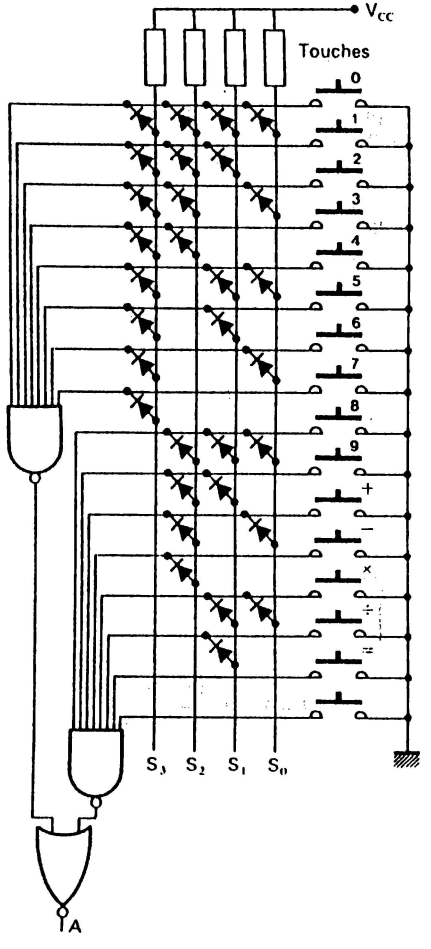


Fig. 9. 1

2. CODAGE DES TOUCHES PAR MULTIPLEXEUR

Dans le système précédent on peut considérer que le dispositif d'enregistrement des données est « passif », attendant qu'une touche soit enfoncée pour effectuer la mise en mémoire.

On peut aussi donner un rôle actif à ce dispositif en lui faisant faire une auscultation permanente des 16 fils de touche.

Imaginons cette fois que ces fils, normalement au niveau haut, soient mis à la masse par l'enfoncement d'une touche. Un dispositif d'interrogation permanente, analogue à un contact tournant (*fig.* 9. 2) détectera le fil qui est à la masse et transmettra l'information 0 sur la sortie sous forme d'une brève impulsion.

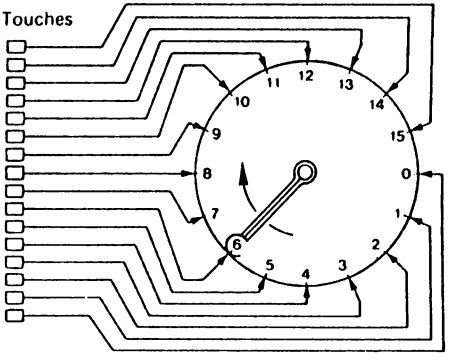


Fig. 9. 2

Il faudra ensuite que le dispositif identifie la touche enfoncée, traduise l'information en binaire et évite de répéter cette information tant que la touche ne sera pas relâchée. Nous allons analyser toutes ces fonctions et étudier les solutions y répondant.

2.1 Détection de l'enfoncement d'une touche

Cette fonction sera assurée par un multiplexeur « seize voies dans une » commandé par un compteur à 4 bits, tel que le circuit 74150. Pour assurer l'interrogation permanente des touches il suffit de brancher une horloge commandant elle-même un compteur de 0 à 15 qui fonctionnera en permanence (*fig.* 9. 3).

La sortie du 74150 étant inverseuse, dès qu'une touche sera enfoncée un créneau positif apparaîtra en sortie.

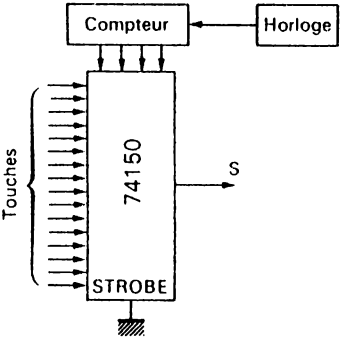


Fig. 9. 3

Si le rythme d'auscultation des touches est suffisamment rapide, l'horloge ayant par exemple une fréquence de 100 kHz, il n'y a aucun risque qu'une touche enfoncée passe inaperçue : aussi rapide qu'il soit, l'homme ne peut maintenir cet enfoncement moins longtemps que quelques centièmes de secondes, le temps de quelques centaines d'interrogations du système.

2.2 Codage en binaire de la touche

Une touche étant enfoncée, le créneau apparaissant sur la sortie S correspond à la valeur binaire affichée par le compteur qui commande le multiplexeur.

Ces 4 signaux (C_0 à C_3) constituent donc le code binaire de la touche.

Il faut donc prévoir un registre qui met en mémoire la valeur de ces 4 bits au moment même où le créneau apparaît sur la sortie.

Le chargement par le créneau même présente un risque : comme ce chargement aura lieu sur le front montant ou sur le front descendant de ce créneau, il surviendra au moment où les données du compteur changent.

Dans ce cas, la solution consiste à organiser le chargement au milieu du créneau par le signal d'horloge inversé, suivant le schéma de la figure 9. 4 *a*, en utilisant une bascule D. Le chronogramme de la figure 9. 4 *b* montre comment s'exécute ce chargement.

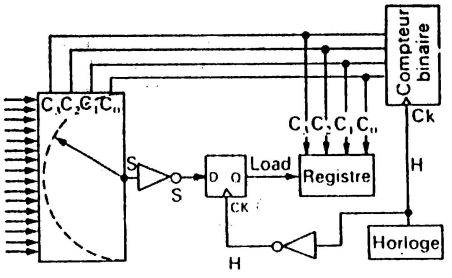


Fig. 9. 4 a

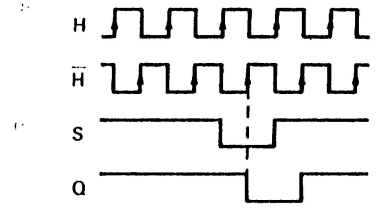


Fig. 9. 4 b

2.3 Non-répétition de l'information

Tant que la touche n'a pas été relachée il n'y a pas de nouvelle donnée à entrer.

Ici il ne sera pas nécessaire, comme pour le cas du codage direct, de créer un signal de commande supplémentaire indiquant que la touche est enfoncée. Ce signal existe en effet sous la forme du créneau de sortie du multiplexeur.

Une bonne solution consiste alors à inhiber le système d'interrogation dès l'enfoncement de la touche pour le remettre en route au moment du relâchement.

Une porte ET commandée par ce créneau (inversé dans le cas du 74150) suffira pour cela (*fig.* 9. 5).

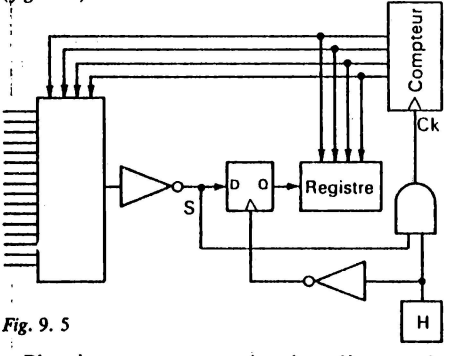


Fig. 9. 5

Plus rien ne se passera alors jusqu'à ce que la touche soit relâchée. En particulier $C_0C_1C_2$ et C_3 gardent leur valeur.

2.4 Système d'exploitation du clavier

Jusqu'à présent nous avons obtenu qu'une valeur binaire déterminée par la touche enfoncée soit enregistrée dans un registre intermédiaire.

Au delà il faut un système d'exploitation qui charge cette valeur en mémoire, incrémente ensuite le compteur d'adresses de la mémoire pour ranger la valeur suivante... à condition que ce ne soit pas la même qui se trouve toujours dans le registre intermédiaire.

Ce système devra donc, à intervalle régulier, vérifier s'il y a ou non une donnée nouvelle à placer en mémoire. Il lui faut pour cela une valeur de contrôle, que nous appellerons L ou « demande de lecture ». Si L est à 1, la valeur enregistrée doit être placée en mémoire, si L = 0 il faut attendre une nouvelle valeur. Un tel bit, servant d'indicateur s'appelle un *drapeau* :

drapeau hissé (L = 1) : valeur à lire,
drapeau baissé (L = 0) : ne pas lire.

Or le chargement du registre intermédiaire a lieu lorsque la sortie Q de la bascule D passe à 0 (*fig.* 9. 5).

L peut correspondre à la sortie $\overline{Q}$ à condition que, lorsque le système a effectivement lu le registre, il envoie un ordre de mise à 0 de L.

Nous organiserons donc un nouveau circuit constitué d'une nouvelle bascule D_1 (dont D_1 est au niveau haut) générant L en sortie lorsque Q passe au niveau bas, donc $\overline{Q}$ au niveau haut ($\overline{Q}$ étant branché sur l'horloge de cette bascule pour une bascule 7474 par exemple). Cette bascule sera remise à 0 par un « Clear » lorsque la mise en mémoire a été effectuée (*fig.* 9. 6).

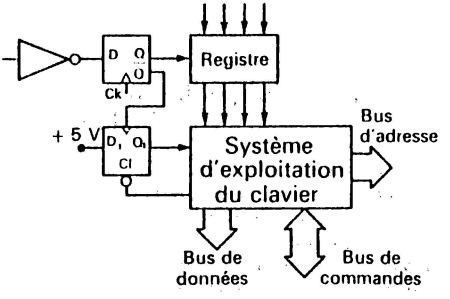


Fig. 9. 6

Nous ne décrirons pas plus en détail ce système d'exploitation qui est complexe. Notons que pour qu'il range les données du clavier en mémoire il doit recevoir et identifier des informations telles que « ceci est une adresse », « ceci est une donnée », donc il doit être relié à certaines commandes du bus de commandes ainsi qu'au bus d'adresses et au bus de données.

3. CODAGE DES CARACTÈRES ALPHANUMÉRIQUES

Un ordinateur ou un micro-ordinateur dispose d'un clavier analogue à celui d'une machine à écrire. En effet, les traitements d'information n'ont pas uniquement trait aux chiffres et aux nombres et, dans de multiples applications, la mise en mémoire et le traitement des caractères alphabétiques est au moins aussi important, sinon plus, que les calculs arithmétiques.

Il faut alors coder les 26 lettres de l'alphabet, les 10 chiffres et divers signes d'opérations plus tout les signes habituels de l'écriture, de la virgule aux guillemets et parenthèses. Pour peu que l'on veuille y rajouter la distinction entre majuscules et minuscules et quelques codes qui seront utiles dans les commandes internes de l'ordinateur, on obtient près de 100 signes, ce qui exige un codage sur 7 bits.

En fait nous avons vu que la norme, en matière de dimensions de mots en mémoire est aux multiples de 2. Ce seront donc 8 bits qui seront utilisés, le bit de poids le plus fort étant utilisé comme *bit de parité*, ce qui permet des contrôles d'erreur.

Ce bit aura une valeur 0 ou 1 telle que l'ensemble des bits 1 de l'octet (8 bits) soit en nombre impair (ou bien en nombre pair pour certains constructeurs).

Des codifications internationales ont évité ici que chaque constructeur n'invente son propre code.

Les deux codes de caractères les plus répandus sont le code ASCII et le code EBCDIC. Nous ne présenterons que le premier.

3.1 Le code ASCII

Son nom est formé des initiales de « American Standard Code for Information Interchange ».

Il est utilisé par tous les fabricants de mini et micro-ordinateurs.

Si l'on excepte le bit de parité, ses 7 bits offrent 128 combinaisons, codées en binaire de 000 0000 à 111 1111 ce qui peut s'écrire, en hexadécimal, 00 et 7F.

Le tableau figurant en annexe donne la correspondance entre les codes hexadécimaux et binaires de l'ASCII et leur signification en clair. On y voit que les 32 premières combinaisons sont consacrées à des symboles particuliers pouvant être utilisés dans la gestion interne des ordinateurs. La signification de ces symboles est donnée au-dessous du tableau.

Annexe Code ASCII
Équivalence hexadécimale et signification en décimal, alphabétique ou symbolique

Hexa-décimal	Binaire	ASCII	Hexa-décimal	Binaire	ASCII
00	000 0000	NUL	17	001 0111	ETB
01	000 0001	SOH	18	001 1000	CAN
02	000 0010	STX	19	001 1001	EM
03	000 0011	ETX	1A	001 1010	SUB
04	000 0100	EOT	1B	001 1011	ESC
05	000 0101	ENQ	1C	001 1100	FS
06	000 0110	ACK	1D	001 1101	GS
07	000 0111	BEL	1E	001 1110	RS
08	000 1000	BS	1F	001 1111	US
09	000 1001	HT	20	010 0000	SP
0A	000 1010	LF	21	010 0001	!
0B	000 1011	VT	22	010 0010	»
0C	000 1100	FF	23	010 0011	#
0D	000 1101	CR	24	010 0100	$
0E	000 1110	SO	25	010 0101	%
0F	000 1111	SI	26	010 0110	&
10	001 0000	DLE	27	010 0111	'
11	001 0001	DC1	28	010 1000	(
12	001 0010	DC2	29	010 1001	)
13	001 0011	DC3	2A	010 1010	*
14	001 0100	DC4	2B	010 1011	+
15	001 0101	NAK	2C	010 1100	,
16	001 0110	SYN	2D	010 1101	-

Hexa-décimal	Binaire	ASCII	Hexa-décimal	Binaire	ASCII
2E	010 1110	.	57	101 0111	W
2F	010 1111	/	58	101 1000	X
30	011 0000	0	59	101 1001	Y
31	011 0001	1	5A	101 1010	Z
32	011 0010	2	5B	101 1011	[
33	011 0011	3	5C	101 1100	\
34	011 0100	4	5D	101 1101	]
35	011 0101	5	5E	101 1110	Λ
36	011 0110	6	5F	101 1111	—
37	011 0111	7	60	110 0000	'
38	011 1000	8	61	110 0001	a
39	011 1001	9	62	110 0010	b
3A	011 1010	:	63	110 0011	c
3B	011 1011	;	64	110 0100	d
3C	011 1100	<	65	110 0101	e
3D	011 1101	=	66	110 0110	f
3E	011 1110	>	67	110 0111	g
3F	011 1111	?	68	110 1000	h
40	100 0000	@	69	110 1001	i
41	100 0001	A	6A	110 1010	j
42	100 0010	B	6B	110 1011	k
43	100 0011	C	6C	110 1100	l
44	100 0100	D	6D	110 1101	m
45	100 0101	E	6E	110 1110	n
46	100 0110	F	6F	110 1111	o
47	100 0111	G	70	111 0000	p
48	100 1000	H	71	111 0001	q
49	100 1001	I	72	111 0010	r
4A	100 1010	J	73	111 0011	s
4B	100 1011	K	74	111 0100	t
4C	100 1100	L	75	111 0101	u
4D	100 1101	M	76	111 0110	v
4E	100 1110	N	77	111 0111	w
4F	100 1111	O	78	111 1000	x
50	101 0000	P	79	111 1001	y
51	101 0001	Q	7A	111 1010	z
52	101 0010	R	7B	111 1011	{
53	101 0011	S	7C	111 1100	|
54	101 0100	T	7D	111 1101	}
55	101 0101	U	7E	111 1110	~
56	101 0110	V	7F	111 1111	DEL

Signification des symboles du code ASCII

Nul	Nul
SOH	Début d'en-tête
STX	Début de texte
ETX	Fin de texte
EOT	Fin de transmission
ENQ	Interrogation
ACK	Acquittement
BEL	Sonnerie ou Alarme
BS	Espacement arrière
HT	Tabulation horizontale
LF	Changement de ligne
VT	Tabulation verticale
FF	Alimentation papier
CR	Retour de chariot
SO	Changement de type de caractères
SI	Changement de type de caractères
DLE	Échappement de liaison de données
DCI	Contrôle dispositif 1
DC2	Contrôle dispositif 2
DC3	Contrôle dispositif 3
DC4	Contrôle dispositif 4
NAF	Acquittement négatif
SYN	Synchronisation au repos
ETB	Bloc de fin de transmission
CAN	Annuler
EM	Fin de support
SUB	Remplacer
ESC	Échappement
FS	Séparateur de fichiers
GS	Séparateur de groupes
RS	Séparateur d'enregistrements
US	Séparateur d'unités
SP	Espace
DEL	Effacement

4. GESTION D'UN CLAVIER ALPHANUMÉRIQUE

Le dispositif de multiplexage adopté pour le clavier de 16 touches n'est plus adapté au clavier de 60 touches (et 120 signes car on utilise la même touche pour coder deux valeurs en se servant de touches supplémentaires telles que « contrôle » ou « shift »).

Il faudrait un multiplexeur 128 en 1 qui n'existe pas, donc mettre en cascade 8 multiplexeurs 16 en 1 et un multiplexeur 8 en 1, et scruter toutes les touches à grande vitesse, au risque de scruter aussi d'assez nombreux rebonds.

On préfère alors utiliser un autre système qui nécessite un codage intermédiaire.

4.1 Codage intermédiaire des touches

Le clavier est organisé en matrice (par exemple 16 colonnes et 8 lignes). Des fils de ligne et de colonne se croisent donc au niveau de chaque touche, sans être connectés les uns aux autres. Chacun de ces fils est « en l'air » et aboutit à l'entrée d'un registre TTL. Son potentiel, au repos, est donc à 1.

La frappe d'une touche a pour résultat de mettre en contact le fil de colonne et le fil de ligne qui passent par cette touche.

Tous les fils de colonne (16) étant scrutés les uns après les autres, par un démultiplexeur « un parmi 16 » commandé par un compteur à 4 bits, lorsque la colonne concernée par la touche enfoncée est scrutée, le fil de colonne passe au niveau 0 et ce 0 est transmis en sortie sur le fil de ligne correspondant. La lecture simultanée du registre de sortie des lignes indique alors sans ambiguïté la touche qui a été enfoncée.

Sur l'exemple de la figure 9. 7, la touche X étant enfoncée, lorsqu'un créneau négatif apparaît sur le fil de colonne n° 7, correspondant aux valeurs 0 1 1 1 de DCBA, un 0 apparaît aussi dans un des registres de sortie.

On peut saisir l'apparition de ce 0 par un « drapeau », obtenu en regroupant toutes les sorties dans une porte NAND. Dès que cette sortie est à 1, on doit provoquer le chargement en

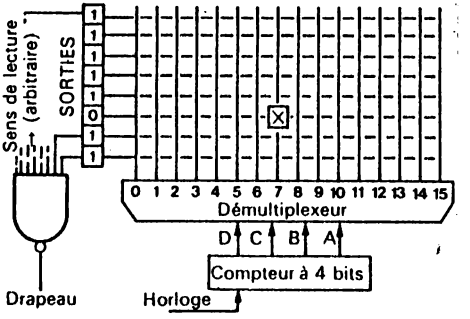


Fig. 9. 7

mémoire des bits DCBA (compteur) qui codent la colonne, et des registres de sortie. On aura alors, d'après l'exemple de la figure 9. 7, la valeur

0111 - 11011111

4.2 Décodage final des touches

On n'obtient donc pas directement le code ASCII par ce procédé. Du reste le code ASCII est indépendant de la position des touches sur le clavier, que ce clavier soit américain ou européen par exemple.

Il va falloir donc utiliser un décodeur, qui pourra être par exemple une mémoire morte.

Le code intermédiaire obtenu servira à adresser cette mémoire, la sortie de mémoire fournira le code ASCII correspondant.

On remarquera toutefois qu'il n'est pas nécessaire d'utiliser 12 bits d'adresse (et 8 192 mémoires de 8 bits) car les 8 bits de droite du code intermédiaire ne représentent pas un nombre binaire, mais bien une position (0) parmi 8. Ce 0 parmi 8 pourra donc servir à adresser un chip de mémoire parmi 8, chacun des chips comportant 16 mots de 8 bits, ce qui représente bien, en tout, 128 mots de 8 bits (7 bits pour le code ASCII plus le bit de parité).

A cela il faudra ajouter le système d'exploitation du clavier. Notons que bon nombre de micro-ordinateurs possèdent leur propre système interne, capable de mettre en mémoire le caractère frappé dès que le code ASCII accompagné du « drapeau » est affiché à leur entrée clavier.

EXERCICE COMMENTÉ

Système d'introduction de données en mémoire par un clavier à quatre touches.

1° Objectif

Il s'agit de réaliser un système pratique d'introduction de données binaires en mémoire par simple frappe sur des touches.

Deux touches correspondant aux valeurs 0 et 1 permettront de composer soit l'adresse où une valeur doit être rangée, soit cette valeur elle-même, chaque nombre (adresse ou donnée à loger en mémoire) étant composé de huit bits.

Une touche « Adresse » permettra d'afficher le nombre composé sur le bus d'adresses de la mémoire.

Une touche « Mémoire » permettra d'écrire le nombre en mémoire.

2° Réalisation

a) Choix du système

Le codage direct des touches tel qu'il est présenté au paragraphe 1 de ce chapitre présente un grave inconvénient lié au rebondissement des contacts. Même avec des touches bien étudiées, il est très difficile d'obtenir un signal pur de tout parasite supplémentaire.

Si l'on observe la figure 9. 1, on aura très difficilement un seul créneau sur la sortie A. C'est pourquoi ce type de solution est généralement abandonné au profit du système de multiplexage. C'est ce dernier que nous adopterons ici.

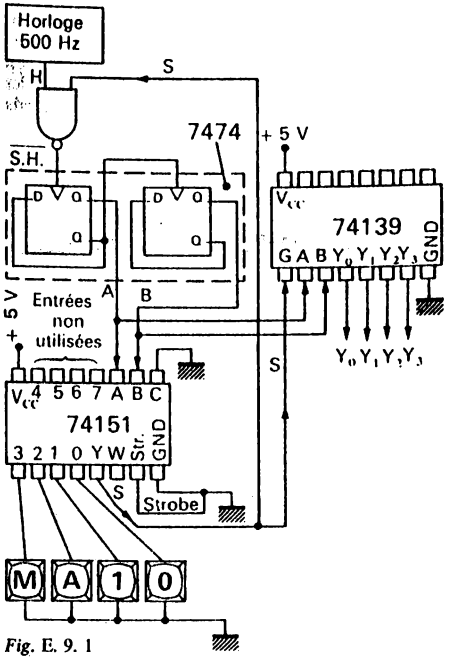


Fig. E. 9. 1

b) Identification des données

Ce système d'identification est fondé sur la solution présentée au paragraphe 2 de ce chapitre.

Une horloge H délivre un signal carré d'une fréquence de 500 Hz. Cette fréquence n'est pas choisie au hasard : elle apparaît optimum parce que suffisamment rapide pour répondre sans problème au frapper manuel des touches, et suffisamment lente pour ne pas trop réagir à des rebondissements intempestifs de ces touches.

Cette horloge active un compteur à deux bascules montées en cascade.

Nous aurons donc quatre valeurs en sortie :

00 affectée à la touche 0
01 affectée à la touche 1
10 affectée à la touche A (Adresse)
11 affectée à la touche M (Mémoire)

Les deux fils de sortie de ce compteur activent un multiplexeur.

Nous avons choisi le circuit 74151 (multiplexeur 8 en 1) dont nous condamnons la moitié des possibilités en mettant la troisième broche de commande C à la masse.

Nous ne nous servirons donc que des entrées 0, 1, 2 et 3, ainsi que de la sortie non inverseuse (Y) que nous appellerons plutôt S.

Les touches elles-mêmes sont branchées entre l'entrée correspondant à chacune d'elles et la masse.

Ainsi, lorsqu'on appuiera sur la touche 1, on verra apparaître en sortie S un signal haut présentant un créneau bas au moment où la valeur 0 1 est présente sur les bornes B et A de commande.

Si aucune touche n'est appuyée, le signal de sortie reste haut.

Nous avons vu aussi qu'il fallait enregistrer en même temps la valeur des deux bits de commande (*a* et *b*) du circuit 151.

Ici, l'enregistrement se fera d'une façon un peu particulière, en utilisant un décodeur (ou démultiplexeur) du type 74139 (double décodeur dont nous n'utiliserons qu'une moitié).

Les entrées de sélection A et B étant respectivement les plots 2 et 3 de ce circuit, nous aurons un niveau bas sur la sortie Y_0 pour l'époque 00, sur Y_1 pour l'époque 01, etc., le reste du temps ces sorties seront à 1.

Il reste à prévoir l'inhibition de la scrutation des touches dès que l'une d'elles est enfoncée. Ceci sera obtenu par une porte NAND recevant le signal d'horloge et la sortie S du multiplexeur.

L'ensemble du système d'identification se présente suivant le schéma de la figure E. 9. 1.

c) Analyse du fonctionnement du système d'identification

Aucune touche n'étant enfoncée, le compteur binaire reçoit l'impulsion $\overline{H}$ (impulsion d'horloge inversée par la porte NAND).

Le multiplexeur scrute en permanence les quatre touches, S reste au niveau haut. Les quatre sorties du décodeur 139 passent à tour de rôle à 0 pour revenir à 1.

Dès qu'une touche est enfoncée, la sortie S du multiplexeur passe à 0 et arrête par la porte NAND le comptage binaire dans la position correspondant à la touche.

Une des sorties du décodeur 139 reste alors au niveau bas, et c'est celle qui correspond à la touche enfoncée.

Comme les sorties de ce décodeur vont nous servir à activer les organes de prise en compte des données, il est indispensable qu'elles n'entrent en action que lorsqu'une touche a été appuyée. Pour cela, le circuit 139 dispose d'une commande d'autorisation ($\overline{G}$) qui inhibe le circuit lorsqu'elle est au niveau haut, bloquant toutes les sorties à l'état haut.

Nous connectons la sortie S à cette entrée $\overline{G}$.

Ainsi tant que le multiplexeur scrute les touches (pas de touche appuyée) le décodeur est inactif et dès qu'une touche est appuyée, une des sorties du décodeur passe au niveau bas et y reste jusqu'à ce que la touche soit relâchée.

d) Changement d'un nombre à 8 bits

Nous allons utiliser pour cela un circuit bien connu, le registre à décalage du type 74195.

Il est bien entendu que seules les touches 0 et 1 sont utilisées pour introduire une valeur binaire.

Si nous devons entrer la valeur 11010010 nous allons tout d'abord appuyer deux fois sur la touche 1 puisque nous lisons de gauche à droite.

Pour que ce nombre soit introduit dans le registre il faut deux conditions successives :

- que la valeur soit présente à l'entrée JK du registre à décalage,
- qu'un front positif d'horloge agisse ensuite pour charger la valeur dans la première bascule, chaque nouveau front positif d'horloge la faisant ensuite déplacer d'une position vers la droite.

Ces deux actions sont réalisées grâce au système présenté en figure E. 9. 2.

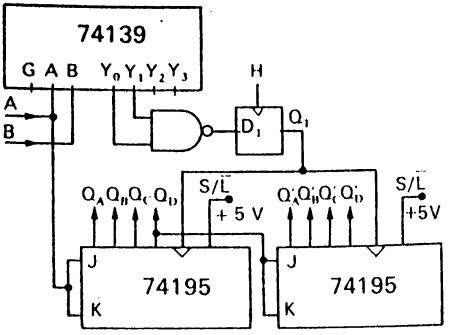


Fig. E. 9. 2

Le fil A en provenance du compteur est directement relié à l'entrée $\overline{JK}$ du registre à décalage. Ce fil porte la valeur 0 si la touche 0 a été enfoncée, et la valeur 1 si la touche 1 a été enfoncée.

C'est l'une ou l'autre des sorties Y_0 ou Y_1 du décodeur qui seront à 0 dans l'un ou l'autre de ces cas. Une porte NAND recevant Y_0 et Y_1 délivrera dans ce cas un signal positif.

Il faut de plus que ce signal soit retardé par rapport à l'instant t_0 où l'enfoncement de la touche a bloqué la scrutation. Pour cela une bascule D, branchée sur H permettra d'obtenir un retard d'une milliseconde.

On obtient le chronogramme de la figure E. 9. 3.

Au bout des huit frappes de touche, le nombre à huit bits est contenu dans le registre à décalage composé de deux 74195 montés en cascade, et il est disponible sur les sorties Q_A à Q'_D.

Remarquons qu'il est rangé cette fois des poids forts à droite, ce qui n'est qu'une considération purement géographique (si le circuit était montré à l'envers, les poids forts seraient bien à gauche). Il sera toutefois bon de savoir, pour tout registre, et en particulier pour les mémoires, où se trouvent les poids forts et les poids faibles.

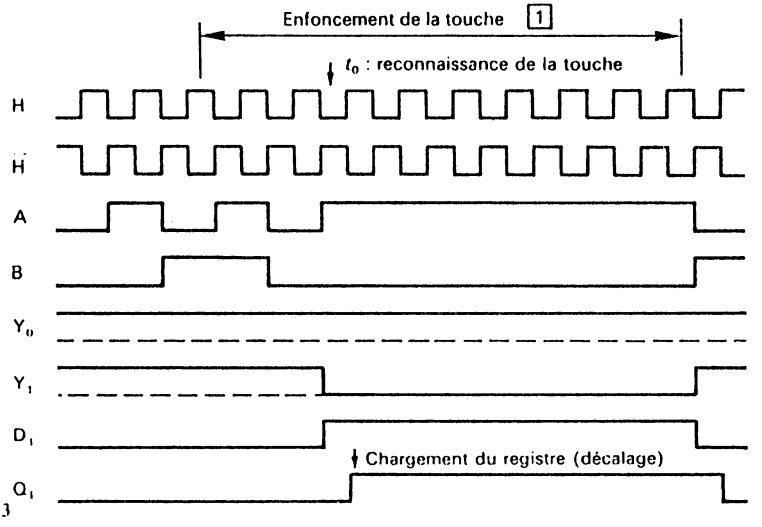


Fig. E. 9. 3

Notons enfin qu'il y a toujours quelque risque de faux rebonds. Il sera donc utile de visualiser le nombre entré par des diodes électroluminescentes placées sur un circuit de type 7416 par exemple, en sortie du registre à décalage. Si une erreur s'est introduite, le plus simple sera de refrapper le nombre.

e) Ordre des opérations

Si nous voulons ranger un nombre en mémoire, il faut tout d'abord adresser le registre de mémoire destiné à contenir le nombre, puis entrer ce nombre. Ceci, en fait, se décompose en quatre opérations :

- frapper le nombre correspondant à l'*adresse*;
- appliquer cette adresse sur le sélecteur d'adresse (bus d'adresses) de la mémoire;
- frapper le nombre correspondant aux données à mettre en mémoire;
- appliquer ce nombre sur le bus de données de la mémoire mise en écriture.

Le premier nombre de 8 bits que nous venons d'entrer est donc supposé être une adresse. Étudions la façon de l'appliquer sur le bus d'adresses en appuyant simplement sur la touche A.

f) Envoi du nombre sur le bus d'adresses

On ne peut pas imaginer d'appliquer directement les sorties Q_A, Q_B ... Q'_D des registres à décalage sur ce bus d'adresse puisque les mêmes registres à décalage doivent aussi recevoir les données à entrer en mémoire.

Il faut donc transférer ces valeurs sur un deuxième registre (RA) dit registre d'adresse (*fig.* E. 9. 4).

On peut, là encore, se servir de circuits 195, dont l'horloge provoquera le chargement (sur front montant de cette horloge) pourvu que la borne « Shift/load » soit à la masse.

L'appui sur la touche A déclenche l'apparition d'un 0 sur la sortie Y_2 du décodeur. Appliqué sur l'entrée D_2 d'une nouvelle bascule activée par H, ce zéro mettra la sortie $\overline{Q}_2$ de cette horloge à 1.

Dès que la touche A est relâchée les entrées de RA sont déconnectées de ses sorties sur le bus d'adresses et on peut introduire les données par les touches 1 et 0. L'adresse, elle, reste affichée sur le sélecteur d'adresse de la mémoire.

g) Introduction de données en mémoire

L'opération est un peu plus complexe et va demander deux temps.

1er temps : Le nombre est introduit dans un registre de données (RM) analogue à RA, mais isolé du bus de données par une porte « trois-états ».

En effet, la mémoire étant adressée, les valeurs qu'elle contient sont présentes sur son bus de données et risqueraient d'entrer en conflit avec les nouvelles valeurs à écrire.

2e temps : Simultanément les valeurs à écrire sont appliquées au bus de données et un ordre d'écriture est donné à la mémoire.

On peut obtenir ces deux temps successifs en jouant sur les phases H et $\overline{H}$ de l'horloge (*cf. fig.* E. 9. 5).

Le signal de Y_3 (touche M) passant à 0, la sortie $\overline{Q}_3$ de la bascule 3 passe à 1 et déclenche le chargement de RM (cette bascule est activée par H).

En cascade avec la bascule 3, une bascule 4 activée par $\overline{H}$ reçoit en D_4 la sortie Q_3 qui vient de passer à 0. Q_4 passe donc à 0 une milliseconde plus tard et vient à la fois :

autoriser la porte « trois-états » d'accès au bus de données de la mémoire,

appliquer l'ordre « Write » sur cette mémoire.

3° Exploitation de la mémoire

Ayant garni la mémoire (capacité 256 mots de 8 bits) on pourra l'exploiter à condition de rajouter un dispositif de visualisation de son contenu, soit par des diodes électro-luminescentes soit par des afficheurs 7 segments.

Il suffira alors de se servir des touches 0 et 1 pour composer l'adresse de la mémoire et de la touche A pour déclencher l'affichage du contenu figurant à l'adresse.

Une telle réalisation, relativement simple, aura permis de se familiariser avec quelques circuits ainsi que quelques principes très largement employés en électronique digitale tels que l'emploi de deux phases d'horloges appliquées à des bascules en série pour obtenir des effets de retardement sans faire appel aux couples RC.

Le schéma d'ensemble est représenté sur la figure E. 9. 6.

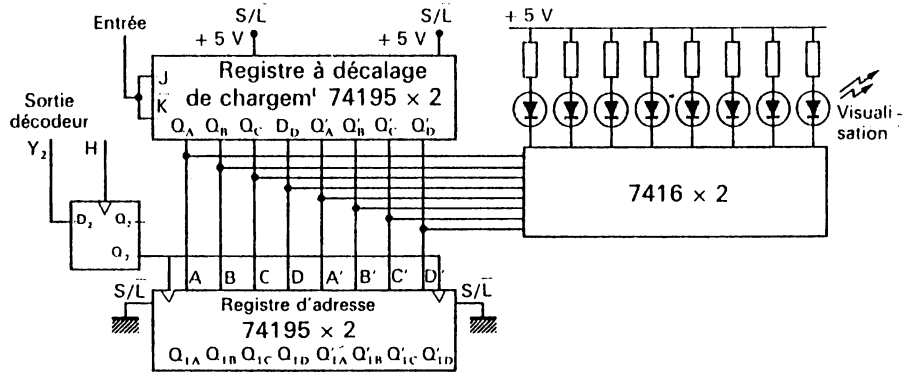


Fig. E. 9. 4

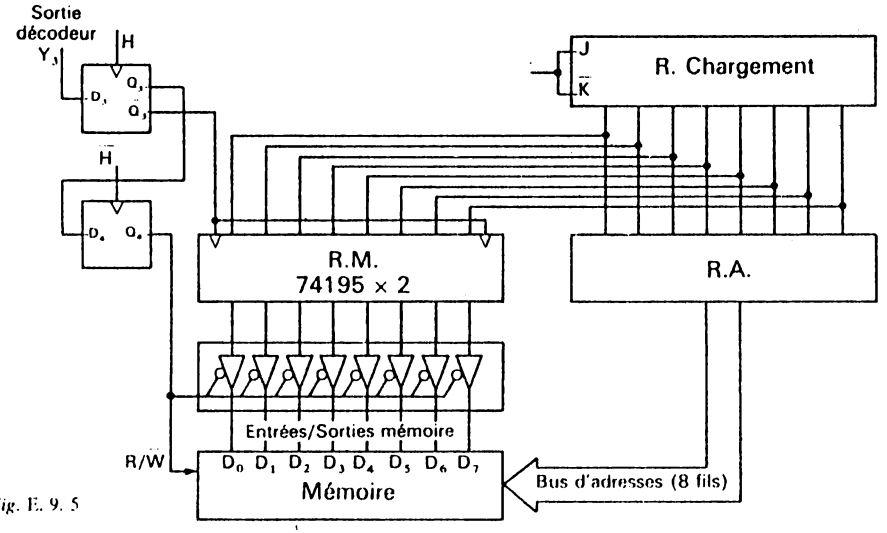


Fig. E. 9. 5

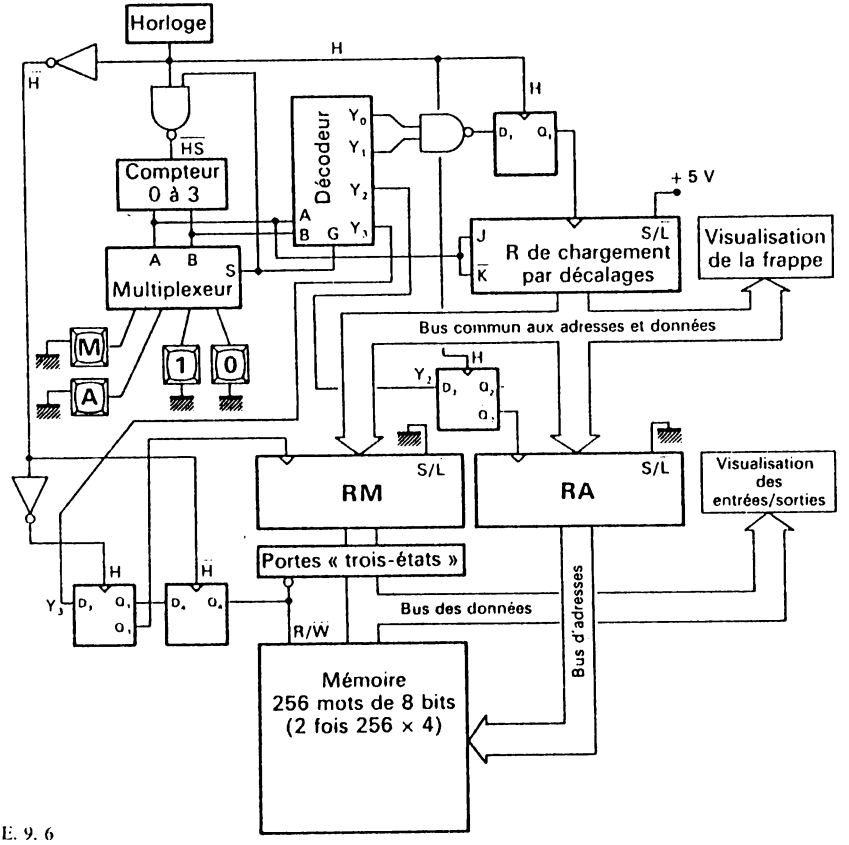


Fig. E. 9. 6

CHAPITRE 10

Au cœur du microprocesseur

INTRODUCTION

Au cours des deux précédents chapitres nous venons de voir

– qu'une machine électronique polyvalente ne peut traiter que des opérations simples; la réalisation d'opérations complexes câblées entraîne des circuits complexes spécialisés,

– qu'entre l'ensemble de traitement des informations et l'homme il y a un relais essentiel qui est la mémoire où seront logées toutes les données transmises par l'homme ou traitées par la machine.

Nous allons étudier, à partir de ces deux considérations essentielles, les principes généraux qui ont conduit à la réalisation des unités de traitement polyvalentes qui, lorsqu'elles sont présentées en «chip» s'appellent microprocesseurs.

1. ARCHITECTURE D'UN MICROPROCESSEUR

1.1 Liaisons avec la mémoire

La mémoire qui est reliée au clavier doit être, bien sûr, reliée aussi au microprocesseur. C'est ce que l'on appelle l'interface essentielle entre l'unité d'entrée qu'est le clavier et l'unité de traitement.

Pour accéder à cette mémoire, que nous appellerons «mémoire centrale» pour éviter toute ambiguïté, le clavier dispose de deux bus : le bus d'adresses pour sélectionner l'adresse d'un mot et le bus de données pour écrire ce mot (ou même pour le lire si l'on dispose en outre d'un moyen de lecture tel que des diodes, des afficheurs, etc.).

De la même façon, le microprocesseur sera relié à cette même mémoire centrale par un bus d'adresses et par un bus de données.

Rappelons-nous encore que le bus d'adresses est toujours uni-directionnel, orienté vers la mémoire, car la mémoire elle-même ne génère pas ses adresses, alors que le bus de données est bi-directionnel : il sort et il entre des données.

Les liaisons entre microprocesseur et mémoire centrale seront donc identiques aux liaisons entre clavier et mémoire centrale.

Comme il ne peut y avoir deux bus d'adresses distincts ou deux bus de données distincts, ce sont les mêmes bus qui sont reliés soit au clavier, soit à l'unité de traitement.

Ceci exige, pour éviter tout accident (court-circuits) d'isoler à tout moment l'un ou l'autre des organes dont la mémoire centrale est l'interface. Cet isolement peut se faire par des portes trois-états par exemple.

Nous aurons alors le schéma de la figure 10. 1.

On notera cependant que de nombreux systèmes comportent la prise en compte de l'exploitation du clavier par l'unité de traitement. Dans ce cas le clavier ne possède pas de registre d'adresses. Sa seule liaison avec le système est constituée par le bus de données.

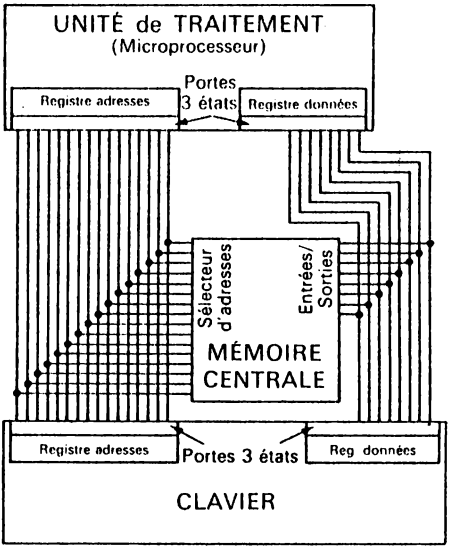


Fig. 10. 1

Dans les exemples que nous donnerons, nous supposerons toujours que la mémoire centrale dispose d'adresses à 16 bits et de mots de 8 bits. Sa capacité est alors de 65 536 mots de 8 bits ou 64 kilo-octets.

Le bus d'adresses aura alors 16 fils et le bus de données 8 fils.

Cette configuration est très répandue. Elle permet de loger dans chaque position mémoire soit un caractère codé en ASCII, soit deux chiffres décimaux codés en BCD, soit un nombre binaire compris entre 0 et 255.

Remarque importante

Pour faciliter l'exposé, nous ne décrirons pas les éléments binaires de chaque mot, tel que 0 1 1 0 1 1 0 1, mais nous en ferons la transcription en hexadécimal ce qui est plus facile et plus court à écrire.

0 1 1 0 1 1 0 1 sera alors appelé $6D_H$ (traduction en hexadécimal des deux groupes de 4 bits).

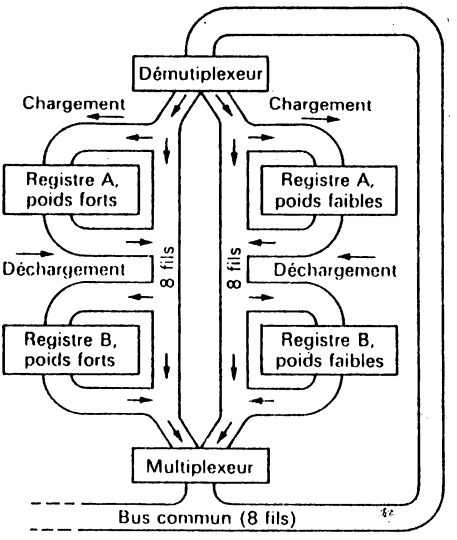


Fig. 10. 2 – Exemple de montage de 2 registres de 16 bits sur un bus de 8 bits (les commandes de sélection, de chargement et de lecture ne sont pas représentées).

1.2 Les outils internes du microprocesseur

1.2.1 Le bus commun

On se rappellera que le clavier (*fig.* E9. 6) comportait un bus interne commun aux adresses et aux données, permettant donc, suivant les commandes utilisées, de transmettre des adresses ou des données.

On retrouvera la même disposition, par une sorte de symétrie, dans le cœur du microprocesseur.

On peut objecter que les données étant sur 8 fils et les adresses sur 16, il y a une incompatibilité de dimensions entre ces deux bus.

La solution passe ici encore par le multiplexage-démultiplexage.

Le bus commun aura 8 fils. S'il s'agit de transmettre une adresse (voir *fig.* 10. 2), cette adresse sera mise dans deux registres de 8 bits dont les contenus seront envoyés l'un après l'autre sur le bus.

C'est un registre d'expédition vers la mémoire centrale qui recevra ces contenus. Il est composé, lui aussi, de deux registres de 8 bits qui recevront l'adresse par l'intermédiaire d'un démultiplexeur.

Ce bus commun interne va, par ailleurs, desservir pratiquement tous les organes internes du microprocesseur. On imagine donc qu'il sera largement doté de portes 3-états et de commandes d'inhibition permettant, à un moment donné, de ne laisser en communication que deux groupes de registres.

1.2.2 L'unité arithmétique et logique

Le cœur du microprocesseur, que nous aurons l'occasion d'étudier un peu plus loin, organise la réalisation de toutes les micro-opérations demandées. Il dispose pour cela d'un séquenceur programmé (cf. chapitre 8) sous la dépendance d'organes tels qu'un compteur et une horloge. On appelle cet organe l'*Unité Centrale de Traitement* (UCT).

Pour réaliser les opérations, il faut que l'UCT dispose avant tout d'un organe de traitement arithmétique et logique performant. Cette *Unité Arithmétique et Logique* (UAL ou ALU en anglais) doit être capable de réaliser, au minimum :

– l'addition de deux octets, avec ou sans retenue,

– la complémentation d'un octet (complément à 1 de chaque bit) pour la soustraction,

– les opérations logiques (AND, NAND, OR, NOR, EXOR) bit par bit sur deux octets,

– le décalage à gauche (multiplication),

– le décalage à droite (division).

A partir de là, on peut considérer que toute opération plus complexe peut être obtenue par une suite de ces opérations simples (comparaison de deux octets par exemple, ou multiplication, ou division etc.).

1.2.3 Les registres internes

Les exercices précédents, et notamment celui du chapitre 9, nous ont montré qu'il était bien utile, chaque fois que l'on traite ou que l'on fait circuler de l'information, de disposer de registres permettant de stocker celle-ci, ne serait-ce qu'un court instant. Ces registres tampons ou verrous (buffers et latches en anglais) doivent donc exister dans le microprocesseur.

Un des plus importants est situé sur l'UAL (*fig.* 10. 3). Il est capable de recevoir l'information résultant du traitement réalisé par cette unité et de la renvoyer à n'importe quel autre registre. Situé donc au carrefour des voies de communication on l'appelle très souvent l'*Accumulateur* (ACC en abrégé). Il peut notamment renvoyer ses informations sur une des entrées de l'UAL. Certains microprocesseurs possèdent deux accumulateurs A et B reliés à la sortie de l'UAL et à l'une ou l'autre de ses entrées.

Nous avons vu aussi qu'il fallait prévoir des registres de stockage intermédiaires pour les adresses, sous formes de doubles registres à 8 bits possédant multiplexeur et démultiplexeur.

Souvent une unité de traitement possèdera plusieurs de ces paires de registres, 4 paires par exemple, adressables par 2 fils de commande et disposant en commun du même système de multiplexage et démultiplexage.

Pour les données sur 8 bits, il sera bon de disposer aussi de registres tampons non jumelés.

Enfin nous ferons mention, plus en détail de deux types particuliers de registres indispensables à la bonne marche de l'unité de traitement : les registres de type LIFO et les registres d'état.

1.2.3.1 Les registres LIFO

Leur nom est composé des initiales de « Last in, first out » qui signifie « dernier entré, premier sorti ». On leur donne aussi le nom de « pile », évoquant l'image d'une pile d'assiettes; l'assiette que l'on retirera en premier est celle qui a été déposée en dernier.

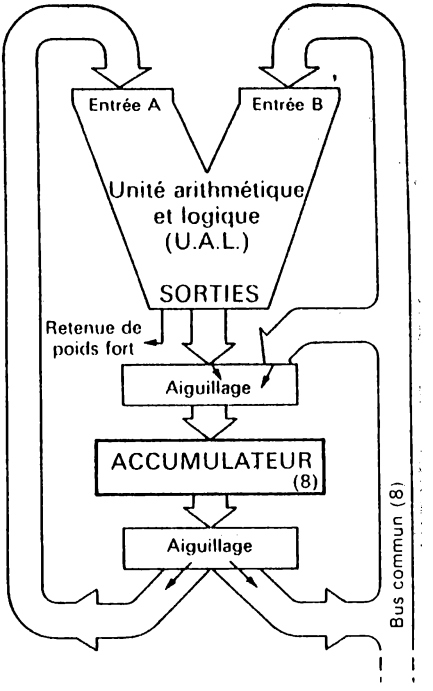


Fig. 10. 3

On peut imaginer la conception de ces mémoires particulières grâce aux registres à décalage.

Si l'on met côte à côte, en parallèle, 8 registres à décalage capables de décaler soit à droite soit à gauche, que les entrées de données soient connectées à l'entrée série de chaque registre (broches JK) et les sorties de données à la sortie Q_A de chaque registre, on voit sur la figure 10. 4 qu'un coup d'horloge sur la commande « décalage à droite » permet d'introduire un

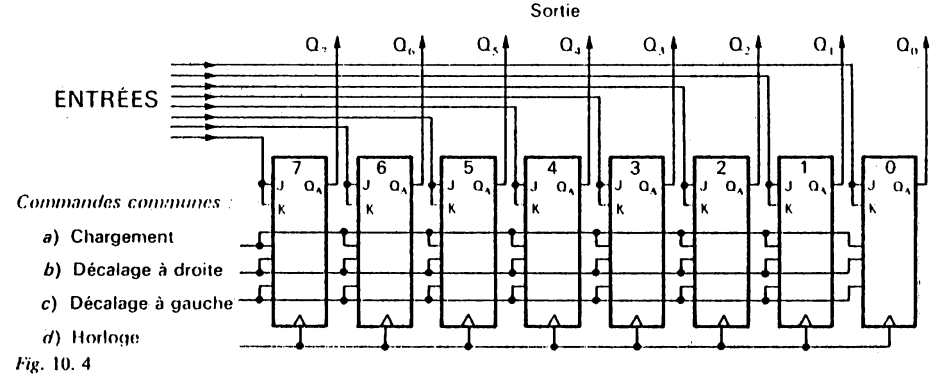


Fig. 10. 4

octet en haut (de la figure) tandis que les octets précédents s'enfoncent vers le bas. A chaque coup d'horloge sur la commande « décalage à gauche » on fera sortir le dernier octet enregistré, puis l'avant-dernier, etc. On a obtenu ainsi une pile LIFO de registres.

1.2.3.2 Les registres d'état

Outre l'unité arithmétique et logique, l'unité centrale aura besoin d'un certain nombre de paramètres pour réaliser ses traitements. Il est indispensable de savoir si l'addition a produit une retenue. Il faudra un registre pour la mettre à la disposition de l'UCT, c'est le registre « Carry ».

Il faudra aussi savoir si le nombre choisi pour diviseur n'est pas nul. Un registre « zéro » détectera cette éventualité. Dans la soustraction nous aurons éventuellement un résultat négatif, d'où l'intérêt d'un registre « Signe », enfin certaines opérations peuvent dépasser la capacité de l'additionneur et de son registre de retenue; c'est un registre de dépassement ou « Overflow » qui le signalera.

1.2.4 Les compteurs

Quelques registres fort importants de l'unité de traitement sont appelés « compteurs ». On s'attendrait donc à ce qu'ils aient, comme tout compteur synchrone (la fréquence des opérations et leur précision ne tolérerait pas de compteurs asynchrones) des dispositifs d'horloge commune à chaque bascule et des liaisons logiques entre ces bascules.

Pourtant, sauf cas exceptionnels, ces registres n'ont de « compteur » que le nom et quelques similitudes de fonctionnement.

Ce sont de simples registres à 8 bits où sont inscrits des nombres. Leur incrémentation ou décrémentation (+ 1 ou − 1) est obtenue de façon programmée. Pour l'incrémentation par exemple :

− le contenu du compteur est envoyé à l'additionneur dont la retenue d'entrée est à 1,

− le résultat apparaît alors sur la sortie de l'UAL et inscrit dans l'accumulateur,

− ce résultat est renvoyé au « compteur ».

Il peut sembler à priori absurde de faire toutes ces opérations pour incrémenter un registre alors qu'une seule impulsion sur l'entrée Ck d'un vrai compteur suffit.

Ce n'est qu'un des aspects du problème plus général de partage entre l'électronique câblée et l'électronique programmée.

Dans le cas du microprocesseur, la réalisation d'un vrai compteur suppose l'intégration de nombreux composants supplémentaires par rapport à une simple juxtaposition de 8 bascules.

De plus on verra que toutes les opérations nécessaires à l'incrémentation ou la décrémentation sont faites en « temps masqué », c'est-à-dire à un moment où, si elles n'étaient pas faites, il faudrait cependant que l'unité centrale attende.

Enfin il faut aussi que ces compteurs puissent, à de fréquentes occasions, voir leur contenu modifié par chargement de nouvelles valeurs n'ayant rien à voir avec les précédentes.

Toutes ces raisons expliquent pourquoi les compteurs de l'unité de traitement sont de simples registres!

En fait les deux compteurs essentiels de l'unité de traitement sont des registres doubles (16 bits), tout comme ceux qui ont été présentés au paragraphe 1.2.3.

Il s'agit du compteur de programme (CP) et du compteur de donnée (CD).

Rappelez-vous bien ces noms pour ne pas faire de confusion entre les deux, d'autant plus que l'un comme l'autre vont emmagasiner des mots de 16 bits qui sont en réalité des adresses.

1.3 Récapitulatif

La figure 10. 5 rassemble schématiquement les composants internes d'un microprocesseur tels que nous venons de les énumérer et de les décrire. On notera le symbole particulier (en V) adopté très généralement pour représenter l'UAL avec ses deux entrées et sa sortie.

Nous n'y avons pas figuré les commandes qui activent les chargements des registres ou leur lecture, la sélection et le fonctionnement de l'UAL, les divers aiguillages du bus commun etc. Ces commandes émanent de l'unité centrale de traitement dont nous allons maintenant étudier le fonctionnement.

2. L'UNITÉ CENTRALE DE TRAITEMENT (UCT)

Si l'on a bien lu les chapitres précédents et réalisés les montages proposés en exercice, bien des notions déjà connues vont servir à comprendre comment est conçue une unité centrale de traitement dans un microprocesseur. Rappelons l'idée générale qui conduit à la polyvalence des traitements possibles : ne faire effectuer à l'UCT que des opérations élémentaires.

C'est l'enchaînement complexe de beaucoup de ces opérations simples qui donne lieu à des développements infinis sous le nom de programmation. Là l'électronicien n'a plus qu'à céder la place à l'informaticien (ou à changer de casquette!).

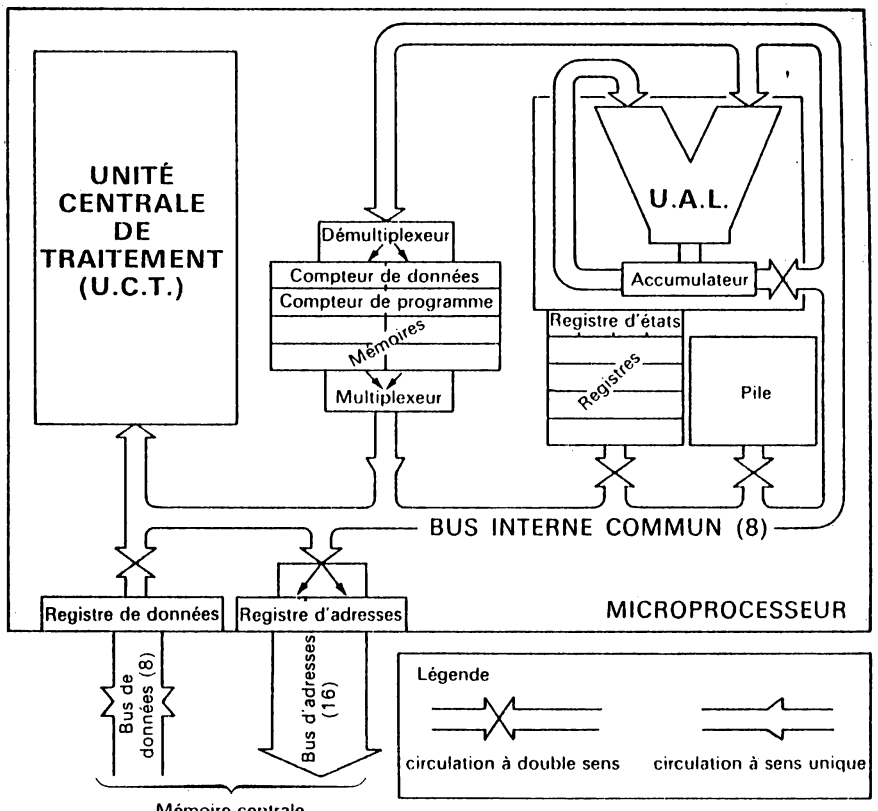


Fig. 10. 5

2.1 Le cycle « fetch »

Dès qu'un microprocesseur équipé de ses moyens minimum de fonctionnement est mis sous tension, il commence à « travailler » et il le fera sans relâche jusqu'à la coupure de l'alimentation.

Mais si aucun ordre ne lui parvient il ne peut que tourner en rond, tout comme le système d'auscultation du clavier étudié au paragraphe précédent. On dit qu'il « reboucle », jusqu'à ce qu'un « drapeau » soit levé, ce qui va lui permettre d'entamer son travail effectif.

Nous passerons les détails de cette initialisation et demandons au lecteur d'admettre qu'à ce moment-là le *compteur de programme* contient une valeur, par exemple 0800_H. Ce nombre est en fait une adresse (elle est codée sur 16 bits) et cette adresse correspond au 1er ordre à exécuter par le microprocesseur dans la suite des ordres que l'utilisateur a inscrits dans la mémoire centrale, de l'adresse 0800 à l'adresse 08A5 par exemple.

Mais il faut encore que le microprocesseur prenne connaissance du premier ordre auquel il doit obéir. C'est pourquoi son activité première doit-elle être d'*aller chercher* cet ordre dont il n'a que l'adresse. C'est le cycle « fetch » (du verbe anglais to fetch : aller chercher).

2.1.1 Cycle minimal

Pour réaliser ce cycle il faut que soient effectuées toutes les micro-opérations suivantes.

a) Transfert de l'octet de poids faible du compteur de programme dans le registre d'adresse, côté poids faible.

b) Transfert de l'octet de poids fort du compteur de programme dans le registre d'adresse, côté poids fort.

c) Envoi d'un ordre de lecture (R) à la mémoire centrale.

d) Transfert de l'arrivée des données de la mémoire centrale (registre de données) sur le bus de données.

e) Chargement d'un registre spécial de l'UCT, dit registre d'instruction.

2.1.2 Cycle « enrichi »

Mais toutes ces opérations n'exigent pas les mêmes délais : quelques dizaines de nanosecondes suffisent pour organiser les transferts internes, du compteur de programme au registre d'adresse par exemple.

En revanche il faut environ dix fois plus de temps entre les opérations *bc* et l'opération *d*, temps nécessaire à la sortie des données de la mémoire centrale à partir du moment où l'adresse est affichée et l'ordre de lecture donné.

Plutôt que de perdre ce temps, il y a possibilité de le mettre à profit pour incrémenter le registre de programme.

Il sera prêt, ainsi, à envoyer la prochaine adresse du programme écrit en mémoire, puisque ce programme est rangé *dans l'ordre dans lequel il doit être effectué*.

Le cycle devient alors

a) Transfert octet faible CP vers RA faible.
b) Transfert octet fort CP vers RA fort.
c) Mise en lecture mémoire centrale.
d) Transfert octet faible CP vers ALU.
e) Commande de l'incrémentation (plus 1).
f) Transfert du résultat vers CP faible.
g) Transfert octet fort CP vers ALU.
h) Commande de l'addition avec la retenue éventuelle de l'opération précédente.
i) Transfert du résultat vers CP fort.
j) Identique à *d*) précédente : transfert du registre de données vers bus interne.
k) Identique à *e*) précédente : transfert bus de données vers RI (registre d'instruction).

2.3 Le rôle du registre d'instruction

Le mot qui vient d'être lu en mémoire et amené dans le registre d'instruction (RI) n'est, bien sûr, qu'une suite de 0 et de 1 qui, à la lecture, ne suggère rien de très particulier. C'est, par exemple, traduit en hexadécimal, le code A9.

Il va représenter pourtant l'ordre à exécuter dans le microprocesseur.

En effet, dès qu'il parvient dans RI, ce code, par un créneau délivré en fin de cycle précédent, est chargé dans le sélecteur d'adresse de la mémoire de microprogramme (MMP) de l'unité centrale.

Ce code est donc une adresse de la MMP et son chargement provoque la lecture de la ligne A9 de cette mémoire ROM (*fig.* 10. 6).

C'est donc bien le contenu du Registre d'Instruction qui sélectionne le programme qui doit exécuter l'ordre. Ce contenu est donc encore une adresse, mais cette fois-ci c'est l'adresse d'un micro-programme. En tant que tel on comprend qu'il n'ait aucune signification en clair. C'est un code, et ce code dépend essentiellement du type de microprocesseur, chaque fabricant ayant agencé ses micro-programmes dans la MMP comme il l'entendait.

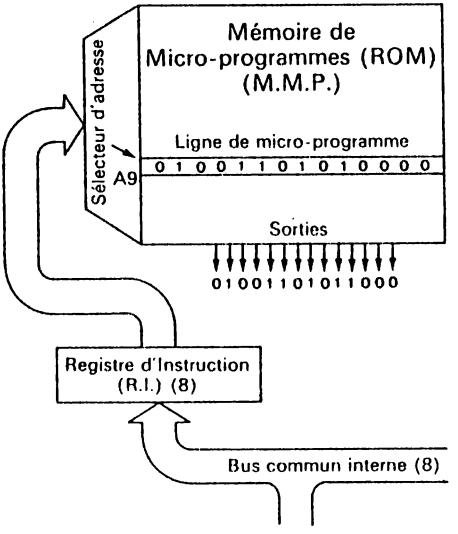


Fig. 10. 6

2.3 Déroulement du micro-programme

Comme nous l'avons vu au chapitre 8, un micro-programme peut comporter de nombreuses lignes de micro-opérations (16 par exemple) qui vont être réalisées chronologiquement.

Une seule ligne de micro-programme n'est donc pas suffisante.

Tous les constructeurs de microprocesseurs n'ont pas adopté les mêmes solutions, et ces solutions vont se modifier avec les nouvelles techniques de très forte intégration qui permettent d'envisager des extensions très importantes de mémoires dans une seule puce (1 024 K-octets par exemple). La solution que nous développerons ici est l'une des solutions possibles.

L'adresse du micro-programme (contenu de RI) n'est pas complète : c'est par exemple l'adresse « haute » d'un mot qui comporte en outre 4 bits d'adresse « basse ».

une horloge et un compteur à 4 bits associés vont donc balayer successivement les 16 registres de la ROM MMP qui ont pour adresse A90, A91,

A92, etc. jusqu'à A9F, ce qui permet de dérouler dans le temps 16 micro-opérations liées à ces 16 lignes de micro-programme (*fig.* 10. 7).

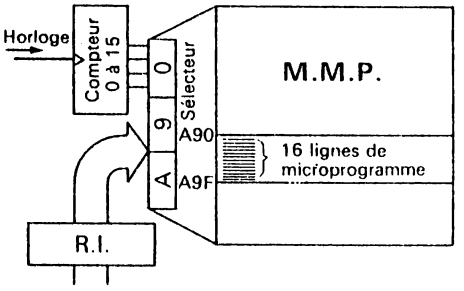


Fig. 10. 7

Chacune de ces lignes est composée d'un nombre de bits lié aux nombres de commandes à envoyer. Chaque bit peut être affecté directement à un fil de commande (éventuellement relayé par une bascule D commandée par l'horloge H ou par l'horloge $\overline{H}$). Une telle solution entraîne des mots très longs. Par exemple, dans les gros ordinateurs, on dispose de mots de 80 bits environ.

La nécessité de restreindre ces mots amène à les grouper en « champs » en les contractant (cf. fig. 10. 8).

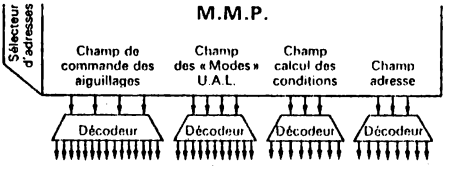


Fig. 10. 8

Si, par exemple, l'unité arithmétique et logique doit fonctionner suivant huit modes différents (décalage, inversion, addition sans retenue initiale, addition avec retenue, etc.) on définira le champ « UAL » grâce à 4 bits qui agiront sur un *décodeur* donnant, en sortie, les huit configurations désirées, sur autant de fils qu'il en faudra (rappelez-vous le principe du décodeur BCD-sept segments).

De même il y aura le champ des aiguillages du bus interne avec les nombreux registres qui lui sont rattachés. Là encore on arrivera, avec 4 bits, à organiser 16 aiguillages différents ou, mieux, 16 configurations différentes d'aiguillages.

Outre ces champs (UAL et aiguillage), le mot de micro-programme va contenir encore un champ de « calculs des conditions » qui prend en compte le contenu des registres d'états pour exécuter ou non telle ou telle micro-opération.

Par exemple, en revenant au cycle « fetch », on voit que les micro-opérations *g* et *h* peuvent être court-circuitées si la retenue de sortie de l'UAL est à 0 lors de l'addition précédente.

Enfin il existe aussi un champ « adresse », fort utile pour aiguiller automatiquement la MMP vers un nouveau programme.

Nous allons en voir un exemple bientôt.

2.4 Chargement d'une donnée

Pour le microprocesseur, l'ordre $A9_H$ signifie, par exemple, « charger dans l'accumulateur la valeur qui suit immédiatement cet ordre ». On appelle ceci un ordre de chargement immédiat.

Comme l'ordre A9 figurait en mémoire centrale à l'adresse 0 8 0 0, la valeur à charger va être logée à l'adresse 0 8 0 1.

L'ordre étant un ordre de chargement en mémoire, le micro-programme d'adresse A9 de la MMP va organiser ce chargement un peu comme dans le cas d'un ordre « fetch », mais au lieu de ranger la nouvelle donnée lue dans le RI, elle la logera dans l'accumulateur.

On se souviendra que le contenu du compteur de programme a déjà été incrémenté lors du cycle « fetch ». Il contient déjà l'adresse 0 8 0 1. Dès lors, la succession des opérations est la suivante.

a) Octet de poids faible de CP vers RA poids faible.

b) Octet de poids fort de CP vers RA poids fort.

c) Mise en lecture mémoire centrale.

d) à *i*) Incrémentation de CP dans l'UAL et reversement dans CP.

j) Transfert du registre de données vers le bus interne.

k) Transfert du bus interne vers l'accumulateur.

On voit que la plupart des opérations sont analogues à celles du cycle « fetch ». En fait elles le sont toutes sauf la dernière.

2.5 Chargement « absolu » d'une donnée

Il est pourtant assez peu fréquent qu'un ordre de chargement soit suivi de la valeur de la donnée à charger.

Lorsque cette donnée résulte en effet d'un calcul précédent effectué par le microprocesseur, le rédacteur du programme n'en connaît pas la valeur.

Il sait en revanche où elle a été logée, c'est-à-dire qu'il connaît l'adresse de la mémoire centrale à laquelle le microprocesseur pourra lire la valeur de cette donnée.

Dans ce cas l'ordre de chargement n'est plus suivi de la valeur (8 bits) mais de l'adresse (16 bits) de cette valeur.

Supposons que cette adresse soit 7000. Alors les registres 0800, 0801 et 0802 contiendront les valeurs suivantes :

0800 : AD (ordre de chargement « absolu » dans l'accumulateur)
0801 : 00
0802 : 70

(Suivant les microprocesseurs c'est la partie basse de l'adresse ou la partie haute qui est chargée en premier, le cas présenté ici étant celui du 6502).

Lorsque l'ordre AD s'affiche sur le sélecteur de la MMP les opérations du cycle « fetch » de a à j vont être exécutées, puis nous aurons :

h) Transfert du bus interne vers la partie de poids faible du compteur d'adresse (CA).

Au cycle suivant, même chose jusqu'à la dernière opération où nous aurons :

k) Transfert du bus interne vers la partie haute du compteur d'adresse (CA).

Au terme du deuxième cycle l'adresse de la valeur figure donc dans le registre d'adresse dont nous voyons là l'utilité.

Mais il reste maintenant à lire la valeur correspondant à cette adresse, et un nouveau cycle d'opérations est à déclencher.

C'est ici que le champ adresse de l'ordre MMP va être utile.

La valeur qui apparaît dans ce champ est transmise, en fin de second cycle, au RI via le bus interne pour venir initialiser le nouveau cycle de micro-opérations :

a_1) octet de poids faible de CA vers RA poids faible,

b_1) octet de poids fort de CA vers RA poids fort.

Ici, CP n'a pas à être incrémenté puisqu'il l'a été lors du cycle précédent et que nous ne faisons pas appel à ce compteur pour recueillir un nouvel ordre.

Il faudra cependant attendre le délai nécessaire pour que la donnée adressée soit stable sur le bus de données externes, quitte à ne pas faire d'opérations pendant ce temps là :

c_1) aiguillage du bus interne vers l'accumulateur,

d_1) à i_1) non-opérations,

j_1) transfert du registre de données vers bus interne et chargement de l'accumulateur,

k_1) voir § 4.

2.6 Gamme des opérations

Selon le microprocesseur (et celui que nous avons imaginé ici est plausible mais ne se réfère à aucun, même si certains codes opératoires choisis font référence au 6502), l'éventail de toutes les opérations réalisables est plus ou moins large.

Tous les microprocesseurs non spécialisés sont capables de réaliser les opérations arithmétiques et logiques parmi lesquelles on peut aussi classer les décalages à gauche ou à droite.

C'est la richesse des possibilités d'adressage qui les distingue surtout. Outre l'adressage immédiat et l'adressage absolu il existe aussi des adressages « indexés » et des adressages « indirects indexés ».

Suivant les microprocesseurs le nombre des opérations différentes (donc des ordres) possibles varie de 50 à 100 ou 120.

Pour sélectionner un de ces ordres il suffit de disposer de 6 ou 7 bits dans le sélecteur de la MMP.

Bien mieux, nous avons vu sur quelques exemples que certains ordres déclenchaient des suites de micro-opérations qui avaient de très nombreux points communs avec d'autres suites de micro-opérations.

Les constructeurs, forcément désireux d'obtenir une MMP la plus compacte possible, n'ont donc pas hésité à utiliser les mêmes lignes de programme en générant directement les variantes par le code opératoire transmis.

On obtient alors un sélecteur d'adresse à six registres, les deux données binaires supplémentaires permettant, au même titre que les bits de micro-opérations, de générer les variantes (*fig.* 10. 9).

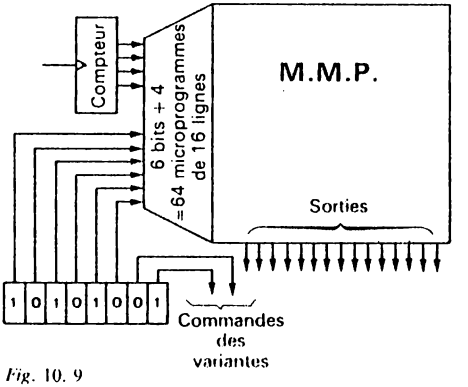


Fig. 10. 9

3. VOCABULAIRE

On appelle *instruction* l'ordre codé qui a été mis en mémoire centrale (dans notre exemple, aux adresses 0800, 0801 et 0802) ou même dans des ROM.

Cette instruction se compose en général de deux parties : le *code opération* (chargement, rangement en mémoire, addition, comparaison...) et *l'opérande*, c'est-à-dire la désignation de la valeur sur laquelle doit porter l'opération ou la désignation de son adresse.

Dans le dernier exemple choisi

AD code opération : « charger l'accumulateur... »

7000 opérande : « ... de la valeur se trouvant à l'adresse 7000 ».

On constate alors que l'appellation « Registre d'instruction » est un terme abusif puisque ce registre ne recevra jamais que le code opération!

On appelle *programme* une suite d'instructions constituant l'ensemble des ordres destinés à faire accomplir par le microprocesseur une suite d'opérations.

Dans le microprocesseur les codes opérations génèrent des micro-programmes qui se déroulent en séquences appelées « *cycles* », chaque élément d'un cycle permettant la réalisation d'une micro-opération.

Remarques :

Certaines instructions ne comportent qu'un code opération (mettre la retenue d'entrée de l'UAL à 0, décaler à droite, à gauche, incrémenter etc.). En fait elles se réfèrent à des circuits bien définis ou des valeurs bien définies qui ont été l'objet de l'instruction précédente. On dit que ces instructions sont à *adressage implicite*.

4. NOMBRE DE CYCLES REQUIS POUR UNE INSTRUCTION

Dès que les micro-opérations nécessaires à la réalisation d'une instruction sont accomplies, le microprocesseur doit aller chercher l'instruction suivante.

C'est pourquoi tout cycle terminal d'une instruction doit aboutir au chargement, dans le registre d'instruction de l'adresse du cycle « fetch », ceci grâce au champ « adresse ».

Comme chaque instruction nécessite au moins un cycle, il faut, en y rajoutant le cycle « fetch » au minimum deux cycles pour réaliser une instruction.

Il en faudra quatre pour faire le chargement absolu, et pour certaines autres six ou sept.

Ceci concerne un microprocesseur déjà performant, tels que les grands classiques de notre époque.

Pour les microprocesseurs modestes, le nombre de cycles peut être beaucoup plus grand, aux dépens de la rapidité d'exécution.

Cette rapidité d'exécution dépend aussi de la fréquence de l'horloge interne. Celle-ci bat souvent à une fréquence 10 à 20 fois supérieure à l'horloge externe et il n'est pas rare qu'un cycle complet se déroule en 500 ns à 1 µs. Il est du reste probable que des progrès seront accomplis encore en ce domaine et que voient le jour des microprocesseurs répondant à des horloges de 100 MHz, à condition que la mémoire centrale soit capable de répondre assez vite à leur interrogation ou leur ordre d'écriture.

5. RÔLE PARTICULIER DE LA PILE LIFO

Nous avons noté, au § 1.2.3.1, l'écriture d'une mémoire interne au microprocesseur d'un type particulier : les données qui y sont introduites « poussent » les données précédemment inscrites vers le bas de la grille. Les données sortent dans l'ordre inverse de leur introduction; les dernières entrées sortent les premières.

Cette pile est utile pour traiter les ordres de « saut à une routine » que nous allons expliquer ici.

5.1 Création d'une routine

Si l'on doit écrire un programme dans lequel il faut prévoir, par exemple, 8 additions successives de deux valeurs, il va être d'une part fastidieux et d'autre part peu économique en places de mémoire d'écrire huit fois à la suite la même série d'instructions. Il est alors possible d'écrire une seule fois ce programme d'addition dans une zone à part de la mémoire de programme. Par exemple, pour un programme allant de l'adresse 0800 à l'adresse 0870, on va placer le programme répétitif en 0880. Ce programme répétitif s'appelle un *sous-programme* ou une routine ou encore « sub-routine » en adoptant le terme anglais.

Chaque fois que l'on voudra dérouler ce sous-programme on écrira l'instruction « sauter au sous-programme » (*fig.* 10. 10).

La dernière instruction de ce sous-programme doit alors renvoyer à l'endroit du programme suivant tout de suite l'instruction de saut. Cette dernière instruction s'appelle « retour au programme principal ».

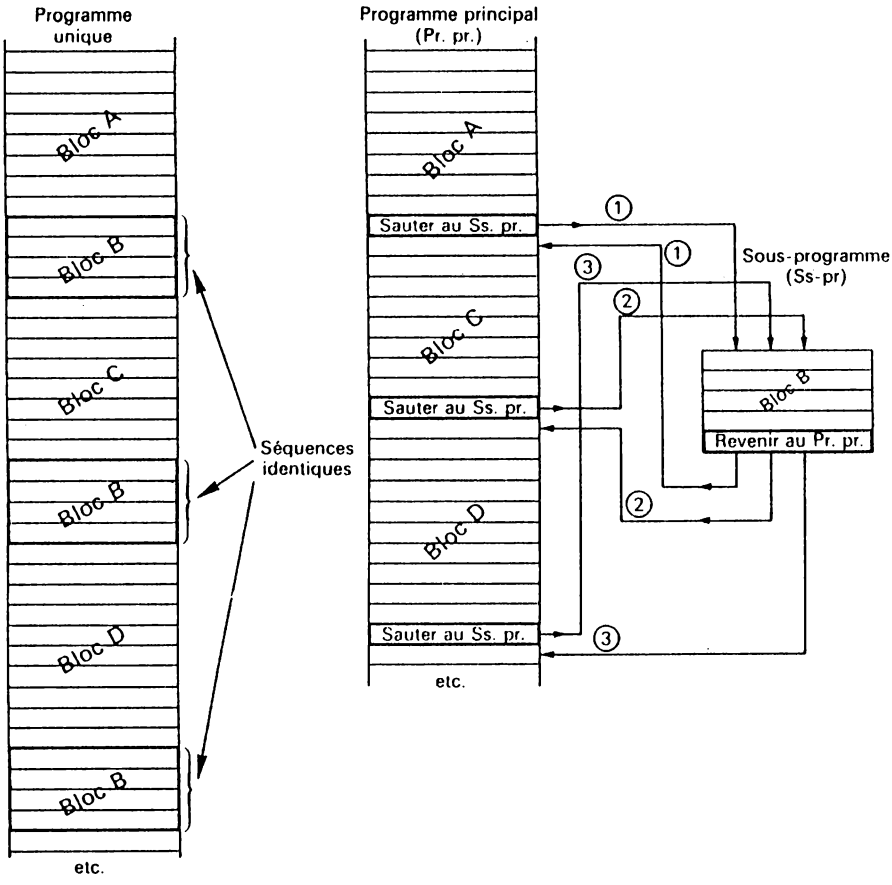


Fig. 10. 10 – Aménagement d'un programme (à gauche) par création d'un sous-programme (à droite).

5.2 Réalisation des programmes de routine

Étudions la façon dont le microprocesseur va exécuter ces instructions.

Si l'instruction « sauter au sous-programme » est écrite en mémoire 080A, et si elle vient d'être enregistrée au RI, les opérations suivantes vont être exécutées :

1er cycle :

– Le CP ayant été incrémenté précédemment, affichage de 080B au RA pour appel du mot d'instruction suivant (adresse basse du saut).
– Incrémentation du CP (080C).
– Lecture de la mémoire centrale et enregistrement dans le CA, poids faible (80).

2e cycle :

– Affichage de 080C au RA pour appel du mot d'instruction suivant (adresse haute du saut).
– Incrémentation du CP (080D).
– Lecture du mot de mémoire centrale et enregistrement dans le CA, poids fort (08).

A ce stade le CA contient l'adresse 0880 à laquelle doit se produire le saut, alors que le CP contient l'adresse 080D où se trouve la suite du programme principal qu'il ne faudra utiliser qu'*après exécution du saut.*

3e cycle :

– Transfert du contenu de CP dans la pile.
– Transfert du contenu de CA dans CP.

Tout est maintenant en place pour que le sous-programme 0880 puisse se dérouler normalement.

5.3 Retour au programme principal

L'exécution de l'instruction « retour au programme principal » est plus rapide car cette instruction est implicite, c'est-à-dire qu'elle n'est pas suivie d'une adresse. Cette adresse est celle qui a été logée au-dessus de la pile (0 8 0 D).

L'instruction « retour au programme principal » se déroule donc très simplement par transfert du contenu du mot supérieur de la pile au registre du CP, octet par octet, ce qui sous-entend l'effacement du contenu précédent du CP. A ce stade le programme principal repartira à l'adresse qui suit l'instruction de saut.

5.4 Chaînes de sous-programmes

L'utilité de la pile n'apparaît vraiment que dans le cas où plusieurs sous-programmes doivent s'enchaîner (*fig.* 10. 11), un sous-programme en appelant un autre, lequel pouvant en appeler un troisième etc. (en général jusqu'à 6 pour la plupart des microprocesseurs actuels).

Alors chaque adresse dite de *branchement* au sous-programme donne lieu à l'introduction dans la pile de l'adresse de l'instruction de raccordement. Six adresses de raccordement peuvent ainsi être empilées.

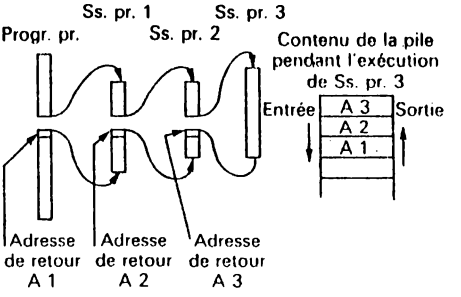


Fig. 10. 11

Lors des retours successifs aux sous-programmes précédents puis au programme principal, le « déstockage » de la pile devra bien se réaliser dans l'ordre inverse du stockage. C'est là le grand intérêt de la pile « Last-in, first-out ».

Remarque :

Certains microprocesseurs ne possèdent pas de pile interne. On prévoit pour eux l'utilisation d'une zone réservée de la mémoire centrale comme pile.

C'est encore un exemple de la possibilité de remplacer un circuit par des ordres supplémentaires, aux dépens, bien sûr, de la rapidité d'exécution. Nous ne développerons pas ici les moyens qui permettent de mettre en œuvre cette simili-pile. On les comprendra mieux à la lecture des chapitres suivants.

CHAPITRE 11

Le langage de programmation des microprocesseurs

INTRODUCTION

C'est grâce à la réalisation d'unités centrales de traitement capables d'exécuter « sur ordre » toute la gamme des opérations élémentaires que l'informatique a pris son essor. Aujourd'hui, avec les microprocesseurs, cette information va pouvoir s'exercer dans tous les domaines d'activité de l'homme, prenant place aussi bien dans ses réalisations les plus sophistiquées (dans le domaine de l'espace par exemple) que dans les outils du quotidien (automobile ou machine à laver par exemple).

C'est qu'une fois franchi le seuil de la réalisation d'unités capables d'effectuer les opérations élémentaires il n'est plus besoin de construire de réseaux logiques complexes pour fournir des ordres à cette unité centrale; il suffit de mettre en mémoire ces ordres sous forme d'instructions codées par des 0 et des 1. C'est le mot-mémoire qui devient ordre et qui est exécuté, et à raison d'un million d'ordres exécutés à la seconde, toutes les qualités de l'ordinateur se révèlent.

Bien que la programmation même dépasse le sujet général de cet ouvrage il nous a paru bon de développer ce qui se trouve à la jonction de l'électronique et des langages habituels de programmation, à la jonction entre ce que l'on appelle le « hardware » (quincaillerie) qui a trait aux réseaux électroniques et le « software » qui n'est plus que langage (du binaire et de l'ASCII aux langages les plus évolués).

Nous allons étudier tout d'abord les instructions, telles qu'elles doivent parvenir à une UCT pour être exécutées.

1. INSTRUCTIONS DE CHARGEMENT DANS L'ACCUMULATEUR

L'étude de ces instructions a un caractère général, applicable à tout microprocesseur 8 bits. Pour lui donner un caractère plus concret, nous l'appliquerons au 6502 (qui équipe notamment les micro-ordinateurs APPLE). Ce microprocesseur nous a paru, d'une part, relativement répandu, d'autre part bon représentant d'une gamme d'instructions assez large. On pourra, sans difficulté, transposer cette étude à tout autre microprocesseur en se procurant la documentation du constructeur.

1.1 Chargement immédiat

Nous avons déjà eu l'occasion, au chapitre précédent, d'aborder deux types de chargement.

Le chargement immédiat, où l'instruction se décompose en 2 octets.

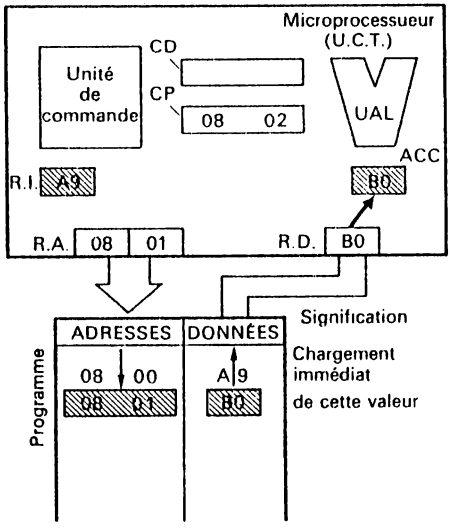
a) Charger dans l'accumulateur,

b) la valeur à charger.

Le code du 6502 est, pour cette instruction, l'octet A9.

Si l'UCT, dans son cycle fetch, affiche donc A9 au registre d'instruction, l'opération qui va se dérouler ne prendra en compte que l'octet suivant, pour le charger immédiatement dans l'accumulateur.

A9 B0 va donc entraîner l'apparition de la valeur B0 dans l'accumulateur (*fig.* 11. 1).



Abréviations :

U.C.T. : Unité Centrale de Traitement
C.D. : Compteur de données
C.P. : Compteur de programme
R.I. : Registre d'Instruction
R.A. : Registre d'Adresses
R.D. : Registre de Données
ACC. : Accumulateur
U.A.L. : Unité arithmétique et logique

Fig. 11. 1 — Chargement immédiat.

1.2 Chargement direct ou absolu

Nous avons vu aussi l'instruction de chargement *suivie de l'adresse* de la valeur à charger. Cette instruction est appelée chargement direct ou absolu, et il ne faut pas la confondre avec un chargement immédiat (sans adresse).

AD 0032 signifie, pour le 6502, charger la valeur (1 octet) *située à l'adresse* 3200 de la mémoire centrale (rappelons que le 6502 lit d'abord l'octet de poids faible de l'adresse, puis l'octet de poids fort; c'est pourquoi on doit inverser les 2 octets de l'adresse) (*fig.* 11. 2 *a*, *b* et *c*).

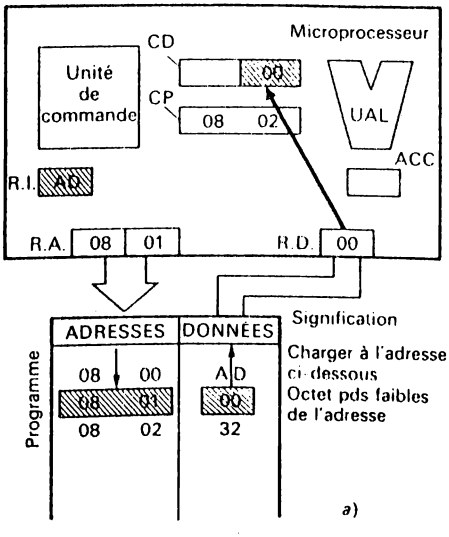


Fig. 11. 2 *a* — Chargement direct ou absolu.

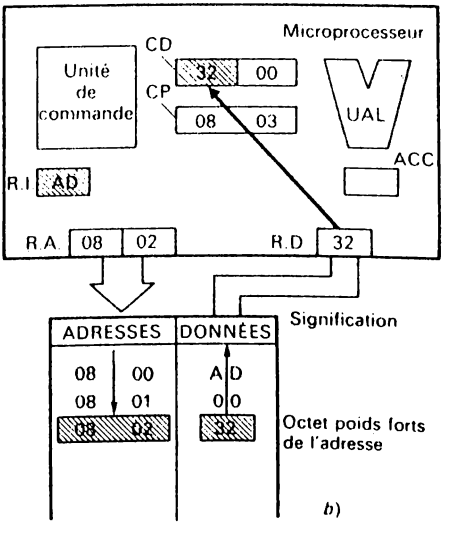


Fig. 11. 2 *b* — Chargement direct ou absolu.

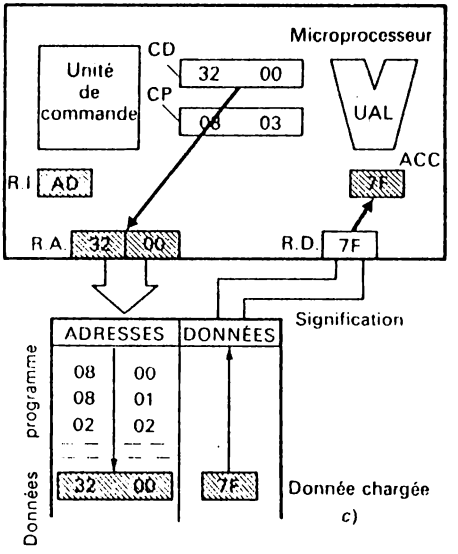


Fig. 11. 2 *c* — Chargement direct ou absolu.

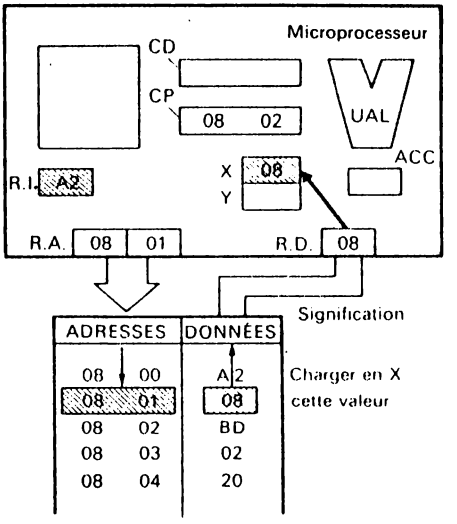


Fig. 11. 3 *a* — Chargement immédiat du registre X.

1.3 Chargement direct (ou absolu) indexé

Nous avons vu qu'un microprocesseur possédait des registres supplémentaires de travail, disposés par huit en général. Parmi ceux-ci le 6502 possède un registre de 8 bits appelé X et un autre appelé Y. On peut donc loger dans X ou dans Y toute valeur de 0 à 255_{10}.

L'instruction A2 permet le chargement immédiat de X et l'instruction A0 le chargement immédiat de Y (cf. *fig.* 11. 3 *a*).

L'instruction BD suivie de 2 octets d'adresse, signifie alors charger l'accumulateur de la valeur située dans la mémoire, correspondant à l'adresse indiquée, augmentée de la valeur de l'index X.

Ainsi BD 0220 (si X = 8) voudra dire charger la valeur située en mémoire 200A (2002 + 8) (voir *fig.* 11. 3 *b*, *c* et *d*).

On peut aussi utiliser le registre Y, dans les mêmes conditions, par l'instruction composée de B9 suivie de l'adresse.

Les deux instructions suivantes :

A0 09 (charger Y de la valeur 9),

B9 0020 (charger l'accumulateur de la valeur située à l'adresse 2000 + Y),

entraînera le chargement, dans l'accumulateur, de la valeur située à l'adresse 2009.

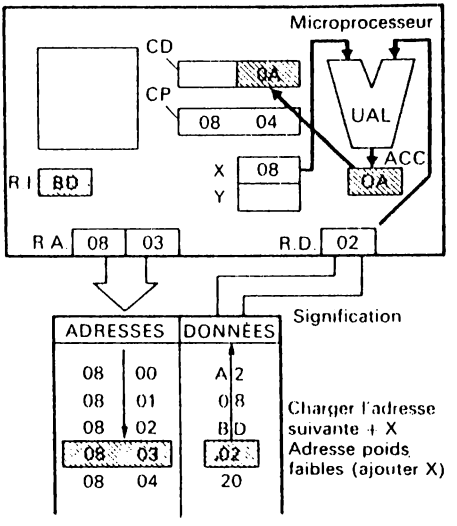


Fig. 11. 3 *b* — Chargement direct indexé (poids faibles de l'adresse).

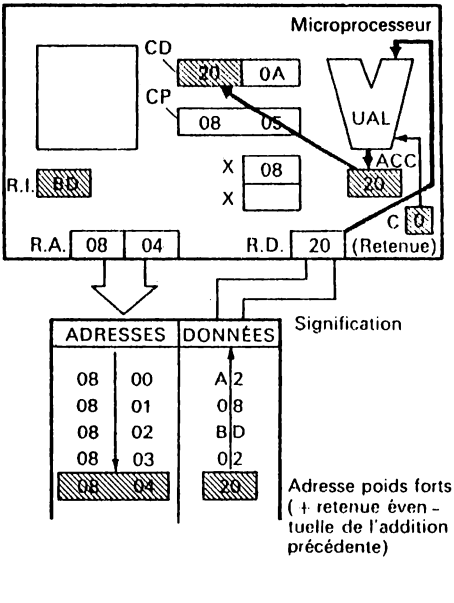


Fig. 11. 3 *c* — Chargement de l'adresse (poids forts).

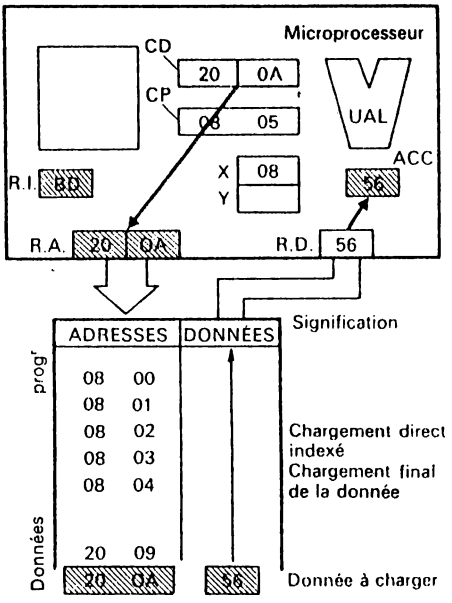


Fig. 11. 3 *d* — Chargement direct indexé. Chargement final de la donnée.

1.4 Chargement « page zéro »

On appelle, de façon imagée, « page zéro » de la mémoire centrale l'ensemble des emplacements dont la partie haute de l'adresse ne comporte que des 0 (les 8 bits de poids forts). Cette page zéro va donc de l'adresse 0000 à l'adresse 00FF.

Pour faciliter l'écriture et la réalisation de programmes, les constructeurs ont souvent prévu des instructions spéciales pour cette page zéro.

Ainsi, pour le 6502, l'instruction A5, suivie d'un *seul octet* d'adresse, demande le chargement d'une valeur à l'adresse « page zéro » indiquée.

A5 A2 signifie charger l'accumulateur de la valeur située à l'adresse 00A2.

1.5 Chargements indirects indexés

Si l'instruction de chargement direct donne l'adresse de la valeur à charger, l'instruction de chargement indirect donne *l'adresse de l'adresse.*

Dans le 6502 ces instructions s'adressent toujours à la page zéro. Le code en est A1 ou B1 pour le chargement en accumulateur.

Ainsi A1 B0 signifie charger en accumulateur la valeur dont *l'adresse est inscrite* en mémoire B0.

Mais B0 ne contient qu'un seul octet alors qu'il en faut deux pour une adresse.

En fait, l'UCT ira chercher le contenu de 00B0 et le placera en compteur d'adresse poids faibles, puis le contenu de 00B1 et le placera en poids forts dans ce même compteur.

Si B0 contient 20 et B1 contient 28, l'UCT ira donc chercher, en fin de compte, le contenu de l'adresse 2820 pour le placer dans l'accumulateur.

Sur le 6502 cet adressage est toujours indexé. Il suffira de faire X = 0 ou Y = 0 pour obtenir le résultat précédent.

Nous allons voir maintenant les effets de l'indexation en commençant par l'utilisation de l'index X.

1.5.1 L'adressage indirect pré-indexé par X (voir *fig.* 11. 4 *a* à *d*)

X intervient comme index au niveau de l'adresse de page 0 qui contient l'adresse terminale.

Si X est porté à la valeur 2, l'instruction A1B0 amènera l'UCT à rechercher, non pas le contenu des adresses B0 et B1 de la page 0 mais les contenus des adresses B0 + 2 et B1 + 2, c'est-à-dire B2 et B3.

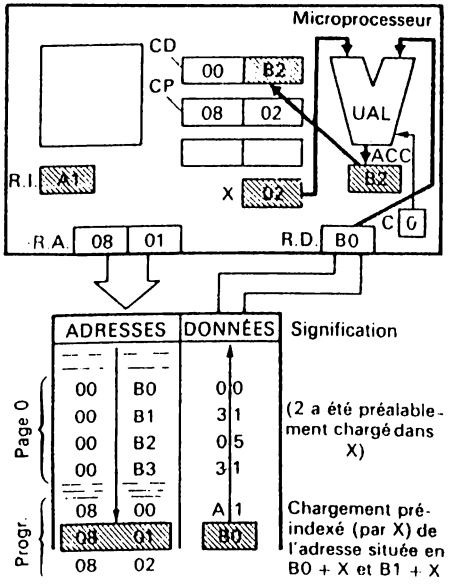


Fig. 11. 4 *a* — Chargement de l'adresse de page zéro, puis modification par X.

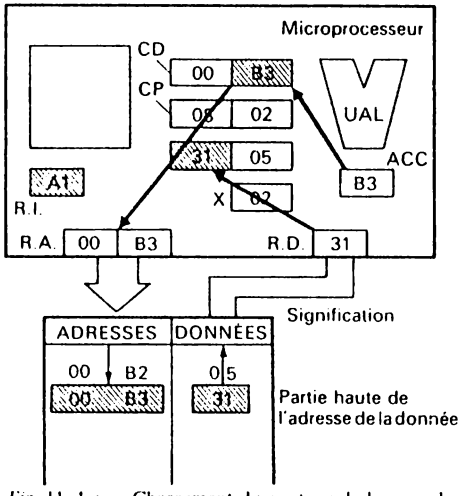


Fig. 11. 4 *c* — Chargement du contenu de la seconde adresse page zéro. Le registre intermédiaire est prêt (il contient l'adresse de la donnée à charger).

1.5.2 L'adressage indirect post-indexé par Y (fig. 11. 5 *a* à *f)*

Avec Y, l'indexation intervient chronologiquement plus tard : une fois que le contenu de la première adresse est entré dans le registre d'adresse.

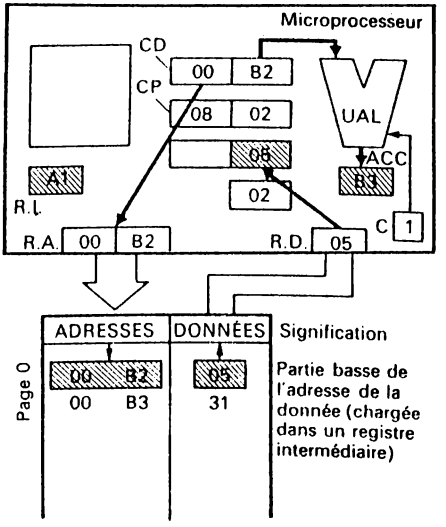


Fig. 11. 4 *b* — Chargement du contenu de la première adresse page zéro, incrémentation de CD.

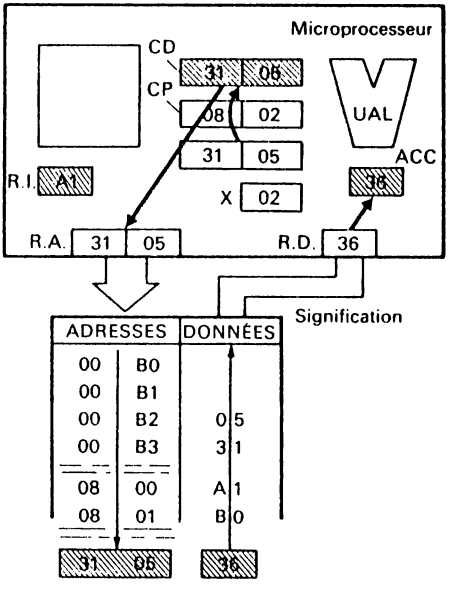


Fig. 11. 4 *d* — Chargement terminal de la donnée.

L'UCT, sur l'instruction B1 A0, va chercher le contenu des registres A0 et A1.

Si ce contenu est 30 22, l'UCT va donc porter dans son compteur d'adresse la valeur 2230 à laquelle *il ajoutera la valeur de l'index Y*, pour charger finalement le contenu de l'adresse 2230 + Y.

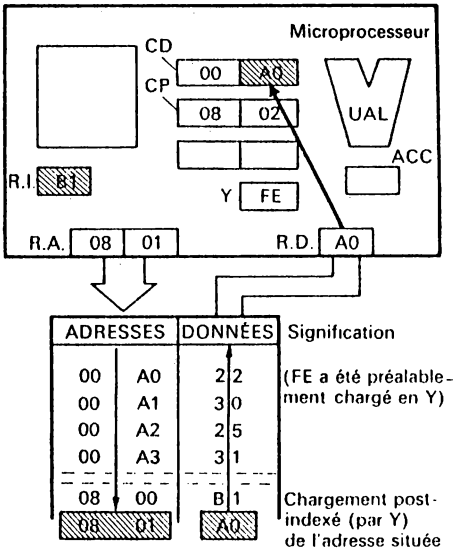


Fig. 11. 5 *a* — Chargement de l'adresse de la page zéro.

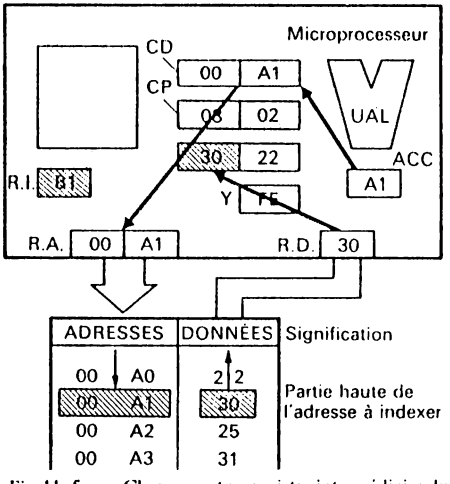


Fig. 11. 5 *c* — Chargement en registre intermédiaire du contenu de la deuxième adresse page zéro.

2. OPÉRATIONS ARITHMÉTIQUES ET LOGIQUES PORTANT SUR DEUX VALEURS

Ces opérations présupposent qu'une première valeur a été chargée dans l'accumulateur. L'ordre d'opération logique interviendra à propos de la deuxième valeur.

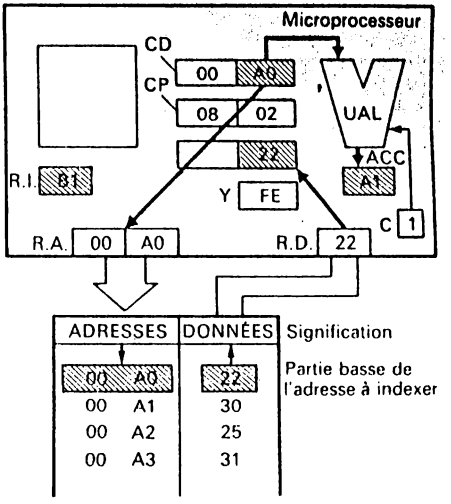


Fig. 11. 5 *b* — Chargement en registre intermédiaire du contenu de la première adresse page zéro. Incrémentation pour l'obtention de la deuxième adresse page zéro.

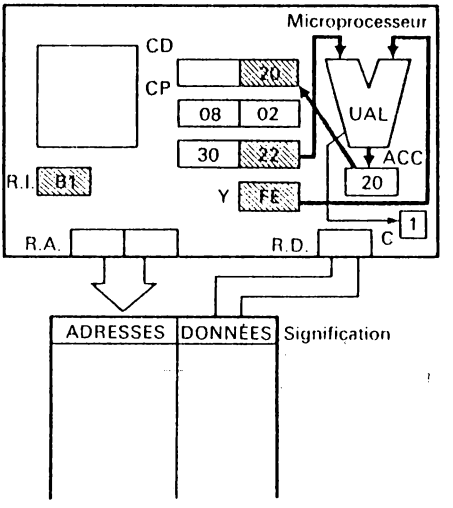


Fig. 11. 5 *d* — Obtention de la partie basse de l'adresse de donnée par addition interne de l'adresse recueillie et du contenu de Y. L'addition génère une retenue.

Cette deuxième valeur peut être :

- additionnée ou soustraite,
- comparée à la première,

ou bien ces 2 valeurs peuvent être l'objet d'opérations logiques, bit à bit, qui sont

- le AND,
- le OR,
- l'EXOR.

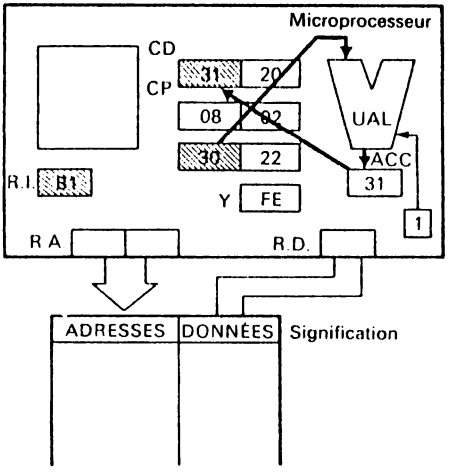


Fig. 11. 5 *e* – Obtention de la partie haute de l'adresse de donnée par prise en compte (systématique) de la retenue (éventuelle) de l'addition précédente.

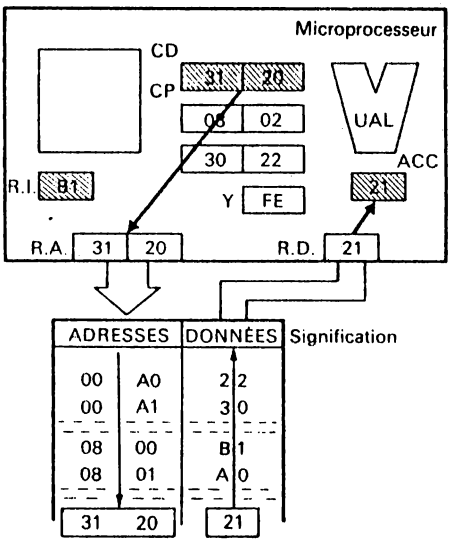


Fig. 11. 5 *f* – Chargement final de la donnée.

Pour ces six opérations, le 6502 dispose, en ce qui concerne la seconde valeur, de tous les systèmes de chargement précédemment décrits, mais l'ordre lui-même sera codé différemment.

Voici un exemple de ces codes pour l'opération d'addition :

	Immédiat	Direct	Page zéro	Absolu X	Absolu Y	Indirect X	Indirect Y
Additionner	69	6D	65	7D	79	61	71

Notons l'utilité de la fonction EXOR pour obtenir la complémentation d'une valeur binaire.

Un premier nombre A étant chargé (B6 par exemple) l'application du nombre FF avec la fonction EXOR va donner 49. En binaire cela se traduit par :

$$\text{EXOR} \begin{cases} 1011\ 0110 & (B6_H) \\ 1111\ 1111 & (FF_H) \end{cases}$$

$$\rightarrow \quad 0100\ 1001 \quad (49_H)$$

Le résultat est bien formé du complément à 1 de chaque bit de la valeur A.

3. OPÉRATIONS D'INCRÉMENTATION ET DE DÉCRÉMENTATION

Parmi les opérations arithmétiques, les additions (+ 1) et les soustractions (– 1) d'une unité ont une importance toute particulière en informatique. Il suffit de songer à l'incrémentation du compteur de programme que nous avons vue précédemment.

Ces opérations sont réalisées par des instructions particulières (lorsqu'elles ne sont pas automatiques comme c'est le cas pour le compteur de programme).

Dans le cas du 6502 des codes d'instruction à adressage implicite permettent :

– d'incrémenter ou de décrémenter X (E8 et CA),

– d'incrémenter ou de décrémenter Y (C8 et 88).

Deux instructions, composées des codes «opération» EE ou CE suivis d'un adressage direct, page zéro ou indirect pré-indexé permettent d'incrémenter ou de décrémenter le contenu d'une mémoire.

On notera qu'il n'y a pas d'instruction particulière pour incrémenter l'accumulateur, puisque dans ce cas une instruction d'addition à adressage immédiat suffit.

N.B. : On notera, pour le 6502, que, sauf instruction préalable, la retenue d'entrée de l'additionneur est à 1. Pour incrémenter l'accumulateur, il suffira donc de l'instruction «additionner la valeur 0».

Si l'on veut par contre obtenir le bon résultat de l'addition de 2 chiffres, il faudra faire précéder l'instruction d'addition de l'instruction « mettre la retenue à 0 ».

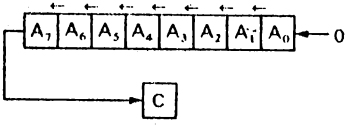
4. OPÉRATIONS DE DÉCALAGE

Il existe plusieurs instructions correspondant à diverses opérations classiques de décalage.

4.1 Le décalage simple à gauche ou à droite du contenu de l'accumulateur

Ces deux opérations peuvent être réalisées avec les codes opération 0A pour le décalage à gauche et 4A pour le décalage à droite.

Dans les 2 cas le bit sortant est placé dans le registre de retenue tandis que le bit entrant est un 0 (*fig.* 11. 6).



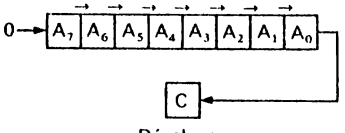


Fig. 11. 6 Décalages

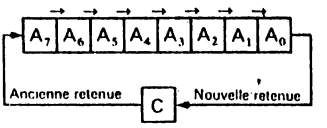
4.2 Décalage de valeurs en mémoire

Des instructions spéciales permettent de réaliser ces mêmes opérations sur des valeurs en mémoire. Le code opération est suivi alors de l'adresse (directe, page zéro, indirect pré-indexé).

4.3 Les rotations

La rotation d'une valeur de 8 bits est ce que l'on pourrait appeler une permutation circulaire d'une valeur composée de ces 8 bits plus le bit contenu dans le registre de retenue.

Pour une rotation à gauche, le bit entrant (poids faible) est le bit de retenue, tandis que le bit sortant (poids fort) est logé en retenue (*fig.* 11. 7).

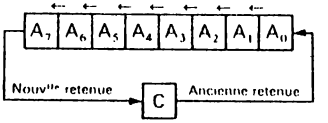


Rotation à droite

Fig. 11. 7

La rotation à droite entraine des effets symétriques (*fig.* 11. 8).

Ces rotations peuvent être commandées par des instructions à adressage implicite (contenu de l'accumulateur) ou, concernant des données en mémoire, avec les systèmes d'adressage des décalages.



Rotation à gauche

Fig. 11. 8

5. LES OPÉRATIONS DE RANGEMENT OU DE TRANSFERT DES DONNÉES

Une fois traitées dans l'UCT, les données, si on ne veut pas les perdre, doivent être rangées à nouveau dans des registres de la mémoire centrale. Ce sont les opérations de rangement ou de stockage (STORE en anglais).

Les instructions de stockage en mémoire de la valeur contenue dans l'accumulateur sont pratiquement aussi variées que les opérations de chargement. Seule y manque l'instruction « stockage immédiat » car en principe on ne stocke que le contenu de l'accumulateur (que l'on n'est pas censé connaître). Si donc l'on veut ranger une valeur précise en mémoire, il faudra d'abord la charger dans l'accumulateur.

De même on peut stocker en mémoire le contenu des registres X ou Y avec des possibilités d'adressage plus limitées (direct ou page 0).

Enfin des instructions à adresse implicite permettent d'effectuer différents transferts :

- de l'accumulateur dans X, dans Y,
- du registre X dans l'accumulateur,
- du registre Y dans l'accumulateur.

6. RÉCAPITULATIF SUR LES INSTRUCTIONS DE TRANSFERT ET DE TRAITEMENT DES DONNÉES : INTRODUCTION DES MNÉMONIQUES

On aura certainement noté, à la lecture des paragraphes précédents, à la fois la multitude d'instructions et de variantes d'instructions que propose un microprocesseur classique et la difficulté à mémoriser le code de chacune de ces instructions ainsi que la nature de l'adressage à associer à ce code. Cette difficulté étant apparue presque insurmontable aux programmeurs (l'usage d'un dictionnaire est toujours possible mais ralentit considérablement le travail), une des premières préoccupations des constructeurs a été de fournir, avec les micro-ordinateurs, des programmes de facilitation de cette programmation. Ce fut la naissance des « codes mnémoniques » c'est-à-dire des codes accessibles à la mémoire humaine.

6.1 Les « mnémoniques » et leur utilisation

Si nous reprenons les instructions de chargement (Load en anglais), nous pouvons facilement retenir, même si nous ne parlons pas couramment l'anglais, que nous appellerons ces instructions « LD », code qui sera suivi de A pour « accumulateur », de X pour le registre X et de Y pour le registre Y.

Nous aurons donc 3 instructions essentielles :

- LDA,
- LDX,
- LDY.

Il reste à marquer la distinction entre les différents modes d'adressage.

L'instruction de chargement immédiat sera composée du mnémonique suivi de la valeur à charger, mais pour ne pas la confondre avec une adresse page zéro qui comporte également un seul octet, nous ferons précéder cette valeur du symbole # (qui peut s'appeler « dièze »).

LDA # 06 = charger l'accumulateur de la valeur hexadécimale 06 (0 0 0 0 0 1 1 0).

L'adressage page 0 sera plus simple : il suffit d'omettre le signe #.

LDA 06 = charger l'accumulateur du contenu de l'*adresse* 0006_H.

L'adressage direct (ou absolu) comportera les 2 octets de l'adresse complète.

LDA 2230 = charger l'accumulateur du contenu de l'adresse 3022 (notez que l'on inverse toujours, sur le 6502, les 2 octets d'adresse).

L'adressage absolu (ou direct) indexé soit par X soit par Y s'écrira sous la forme

LDA 0030, X
LDA 2030, Y

(charger l'accumulateur de la valeur située en adresse 3000 + X, en adresse 3020 + Y).

Enfin les adressages indirects seront écrits sous la forme suivante :

adressage pré-indexé : LDA (A0, X)
(charger dans l'accumulateur la valeur *dont l'adresse* se trouve en mémoires A0 + X et A1 + X)

adressage post-indexé : LDA (B0), Y
(charger dans l'accumulateur la valeur dont l'adresse est celle contenue en B0 et B1, *augmentée* de Y).

6.2 Utilisation des mnémoniques

Comme un micro-ordinateur dispose de tous les caractères alphanumériques, il est simple de frapper ces codes mnémoniques sur ce clavier. Encore faut-il que ces codes soient traduits en machine pour devenir les codes accessibles à l'UCT.

Sans entrer dans le détail des solutions adoptées, nous dirons que la traduction va faire appel à un programme (et non pas du « hardware »), qui fera lui-même ce travail : à chaque ligne d'instruction qui sera frappée, le système renverra à ce programme pour effectuer la traduction en code machine et, éventuellement, transmettre un message d'erreur si l'expression frappée n'est pas conforme à un des codes mnémoniques.

Dans d'autres cas la traduction se fait par blocs entiers d'instructions.

Ces programmes portent le nom d'assembleurs.

Chaque microprocesseur a son propre jeu d'instructions qui diffère de celui des autres microprocesseurs, et chaque constructeur propose aussi ses propres codes mnémoniques. Bien que ces codes soient souvent différents, il est relativement facile de trouver la correspondance entre eux et de passer de l'un à l'autre. Tant qu'il n'existera pas dans ce domaine de code universel, il ne sera pas possible, dans un ouvrage comme celui-ci, de présenter des instructions machines sans faire référence à un type de microprocesseur particulier. Nous poursuivons ce chapitre en présentant les mnémoniques du 6502.

7. LES INSTRUCTIONS DE PRISE EN COMPTE DES CONDITIONS

Les instructions qui ont été étudiées aux paragraphes précédents (transfert de la mémoire vers l'UCT ou de l'UCT vers la mémoire, opérations arithmétiques et logiques) pourraient paraître suffisantes pour assurer la plupart des tâches que l'on attend d'un microprocesseur.

7.1 Les registres d'états

Pourtant bon nombre d'opérations seraient impossibles à réaliser si nous n'avions pas la possibilité de prendre en compte un certain nombre d'éléments et notamment ceux qui sont contenus dans les « registres d'états ».

Rappelons ces principaux registres

C est le registre « retenue ». Il enregistre la retenue d'une addition. En début de programme C est toujours à 1 mais une instruction implicite CLC (Clear Carry) permet de mettre C à 0.

Z est le registre d'état « zéro ». Si le nombre contenu dans l'accumulateur est nul, Z se met à 1. Dans tous les autres cas Z = 0.

N est le registre d'état du « signe ». N contient en fait le bit de poids le plus lourd de l'accumulateur. Lorsqu'une addition avec complément à deux (c'est-à-dire une soustraction) est réalisée, N = 1 indique que le signe du résultat est négatif.

V est le registre de « débordement ». Il est également utile dans les opérations « signées » pour indiquer les risques de débordement. C'est en fait l'avant dernier bit de poids lourd de l'accumulateur.

7.2 Les conditions à respecter

Chaque fois que nous faisons des opérations, nous respectons presque instinctivement certaines conditions. Si nous faisons une multiplication de deux nombres, nous commençons par les unités du multiplicateur, puis nous prenons les dizaines, etc. mais nous nous arrêtons lorsqu'il n'y a plus de chiffre significatif : nous ne tenons pas compte de la série infinie de 0 qui représente tous les poids forts du multiplicateur après son dernier chiffre significatif.

Ou bien si on nous demande de compter de 1 à 10 nous ne partons pas de 0 et nous nous arrêtons à 10.

Le microprocesseur ne sachant faire que ce qu'on lui dit de faire n'a pas ces « instincts ». Si nous voulons voir se terminer les opérations qu'on lui a confiées, il faut à tout moment lui « remettre en mémoire » la raison pour laquelle il doit s'arrêter ou passer à un autre programme. D'où l'importance des « instructions conditionnelles » dites aussi instructions de *saut* ou de *branchement* conditionnelles.

7.3 Les instructions de saut ou de branchement conditionnelles

Les principales instructions, avec leur mnémonique, sont les suivantes :

- Branchement si C = 0
  (BCC) Branch if Carry Clear
- Branchement si C = 1
  (BCS) Branch if Carry Set
- Branchement si Z = 1
  (BEQ) Branch if Equal to 0
- Branchement si Z = 0
  (BNE) Branch if Not Equal to 0
- Branchement si N = 1
  (BMI) Branch if Minus
- Branchement si N = 0
  (BPL) Branch if Plus

Branchement si V = 0
(BVC) Branch if Overflow Clear

- Branchement si V = 1
  (BVS) Branch if Overflow Set

7.4 Exemple simple d'application

Rappelons tout d'abord qu'il existe une opération arithmétique qui s'appelle la comparaison.

Une valeur étant chargée en accumulateur (appelons-la A), l'instruction « CMP + adresse » permet de comparer A à la valeur B située à l'adresse indiquée.

L'UCT fait alors une soustraction de B à A, mais le résultat n'est pas inscrit dans l'accumulateur qui conserve la valeur A; seuls les registres d'états sont modifiés comme si la soustraction avait eu lieu.

Alors

- si A est supérieur à B, Z = 0 mais C = 1,
- si A est égal à B, Z = 1 et C = 1,
- si A est inférieur à B, Z = 0, C = 0.

Pour connaître le résultat de la comparaison, il sera indispensable donc d'utiliser les instructions de branchement, en se rappelant que toute instruction de branchement qui ne détecte pas les conditions requises par l'ordre est comme ignorée par le microprocesseur qui passera à l'instruction *suivante*.

Si, en revanche, les conditions requises sont présentes, le programme sera branché à l'adresse qui suit l'instruction de branchement.

Donc, à la suite de l'instruction CMP + adresse B, on pourra exploiter le résultat de la comparaison par :

- BCC active si A < B,
- BEQ active si A = B,
- BCS active si A ⩾ B.

Pour savoir si A > B, il faudra mettre deux conditions de branchement :

BEQ suivie de BCS.

7.5 Adresses de branchement

Toutes les instructions de branchement conditionnelles du 6502 sont à *adressage relatif*. Ce terme indique que l'adresse qui suit l'instruction est en fait un numéro qui indique *combien d'octets de mémoire il faut sauter* pour arriver à l'instruction sur laquelle doit se brancher le programme.

Ce mode d'adressage, qui peut paraître a priori un peu barbare, est adopté de façon quasi universelle pour des raisons de facilité. La principale de ces raisons est qu'un programme doit pouvoir être rangé en mémoire à une adresse de début ou à une autre. Bien mieux, on a quelquefois à la transférer d'une adresse à une autre.

Si alors les instructions de branchement étaient suivies d'un adressage direct, il faudrait corriger toutes ces adresses. L'adressage relatif évite cet inconvénient.

7.6 Pratique de l'adressage relatif

Soit le programme suivant

(1) LDA 0820
(2) CLC
(3) ADC # 0A
(4) BNE 08

Si ce programme est logé à partir de la mémoire 1001_H nous aurons le rangement suivant

1001 : AD (code de LDA adressage direct)
1002 : 08 (octet bas de l'adresse)
1003 : 20 (octet haut de l'adresse)
1004 : 18 (CLC = effacer la retenue)
1005 : 65 (code de l'addition immédiate)
1006 : 0A (valeur à additionner)
1007 : D0 (BNE = branchement si A ≠ 0)
1008 : 08 (nombre d'octets à sauter).

A quelle adresse le branchement prévu aura-t-il lieu si effectivement le résultat de l'addition n'est pas nul?

Si tel est le cas, lorsque l'octet de la mémoire 1008 est pris en compte, rappelons-nous que le compteur de programme est automatiquement incrémenté, avant même que la valeur 08 n'apparaisse sur le registre de données.

L'instruction BNE 08 va donc aboutir à l'addition de la valeur 8 au contenu du compteur de programme qui est déjà 1009

1009 + 8 = 1011 (en hexadécimal!)

Ainsi toute la partie du programme comprise entre 1008 et 1011 sera ignorée et l'UCT chargera directement l'ordre inscrit en 1011.

Ce saut en avant n'est cependant pas illimité : il est possible jusqu'à la valeur binaire 0 1 1 1 1 1 1 1 qui équivaut à 127 en décimal (7F en hexadécimal).

En effet, dès que l'adressage relatif dépasse cette valeur, le bit de poids le plus fort passe à 1 et le micro-programme de saut est conçu de telle façon qu'il interprète alors ce saut comme un saut *en arrière*.

Si l'instruction de saut est BNE 80, l'addition de 80_H avec la valeur 1009_H devrait donner 1089_H alors qu'en fait l'octet de gauche du résultat est décrémenté de 1 et devient $0F_H$ (0F + 1 = 10).

L'UCT fera donc un saut à la mémoire de programme 0F89.

C'est ainsi qu'un adressage relatif permet de remonter de 127 octets au maximum ou de descendre de 127 octets.

8. SAUTS INCONDITIONNELS

Il existe enfin des instructions de saut sans condition préalable (mais que l'on peut associer à des instructions de branchement). C'est l'instruction JUMP (JMP) c'est-à-dire SAUT et l'instruction Jump à une sous-routine (JSR).

La première (JMP) entraîne l'effacement du contenu du compteur de programme et son remplacement par l'adresse de «jump».

On sait (voir chapitre 10) que le saut à une sous-routine doit entraîner en revanche la sauvegarde de l'adresse précédente du compteur de programme pour permettre le retour au programme principal.

En regard de cette instruction JSR, le programme de sous-routine doit se terminer par une instruction de retour au programme principale. C'est l'instruction RTS (Return from subroutine).

9. FONCTIONNEMENT DE LA PILE DU 6502

Dans le 6502 la pile n'est pas un ensemble de registres internes à l'Unité Centrale mais elle emprunte des mémoires à la mémoire centrale.

Ces mémoires sont ce que l'on appelle la page « 1 » qui suit immédiatement la page zéro, c'est-à-dire les mémoires dont les adresses vont de 0100 à 01FF.

Chaque fois qu'une adresse du CP est mise dans cette pile, elle occupe deux positions de mémoire.

Un index est associé à cette mémoire. Il se trouve, lui, dans l'UCT sous forme d'un registre de 8 bits plus un neuvième registre toujours à 1. Cet index appelé pointeur de pile (stack pointer en anglais, d'où son nom de registre S) enregistre à tout moment l'adresse du « sommet » de la pile.

Des instructions particulières permettent :

– d'entrer une adresse dans la pile, ou plutôt une demi-adresse (il faudra faire l'opération deux fois pour une adresse) par l'instruction PHA implicite : le contenu de l'accumulateur est poussé dans la pile,

– de faire entrer le registre d'états dans cette pile par l'instruction PHP (le registre d'états est aussi appelé registre P).

Cette dernière opération, complétée par l'opération de dépilement dans le registre accumulateur (PLA) permet de mettre en mémoire et de connaître le contenu du registre d'états.

Outre PLA (dépiler dans l'accumulateur) on dispose aussi de PLP (dépiler dans le registre d'états).

Chacune de ces instructions comporte la mise à jour automatique du registre S ou « pointeur de pile ».

EXERCICES

1. Multiplication de deux nombres compris entre 0 et 255

Chaque registre de mémoire pouvant contenir un nombre compris entre 0 et 255, nous allons bâtir un programme permettant de faire la multiplication de deux nombres préalablement introduits dans la mémoire centrale.

Notre exemple sera bâti sur l'organisation du 6502, de ses instructions et de ses mnémoniques (voir chapitre 11).

a) Rangement des deux nombres à multiplier

Toutes les mémoires du micro-ordinateur ne sont pas disponibles. Nombre d'entre elles servent à sa gestion interne, d'autres sont affectées à l'affichage de dessins sur écran par exemple.

Nous choisirons une zone de mémoire disponible qui est la zone 9000_H à $9C00_H$. Il nous suffira du reste de quelques-unes de ces mémoires pour ranger et le programme et les données.

Le programme sera logé à partir de 9000,

les données à partir de 9100.

Imaginons donc de loger en *9101* le premier nombre à multiplier et en *9102* le deuxième.

Le résultat sera au maximum de 65025, c'est-à-dire 1111-1110-0000-0001 en binaire pur. Il nous faudra deux octets pour le ranger. Nous affecterons les mémoires *9103* et *9104* à ce résultat, 9103 étant réservé aux poids forts et 9104 aux poids faibles.

b) Principe du programme

Soit A le nombre logé en 9001 ; B celui de la mémoire 9002 sera considéré comme le multiplicateur.

Nous examinerons successivement tous les bits de B en commençant par la droite.

Si le bit le plus à droite de B est égal à 1, la multiplication de A par 1 donnera un résultat partiel égal à A.

Il faudra ensuite décaler A vers la gauche (comme dans toute multiplication faite à la main) pour préparer le deuxième résultat partiel.

Si le bit suivant de B est 0, le résultat de A × 0 étant 0, nous n'aurons pas de résultat partiel significatif, mais il faudra pratiquer un nouveau décalage vers la gauche sur A.

Plutôt que d'engranger des résultats partiels, nous en ferons l'addition au fur et à mesure.

Exemple : $A = 01011011\ (91_{10})$
$B = 00000101\ (5_{10})$

Première phase : 1er bit de B = 1

$$A \times 1 = 01011011$$

rangé dans la mémoire résultat (R).

Deuxième phase : Décalage de A vers la gauche

$$A_1 = 010110110$$

Troisième phase : 2e bit de B = 0

$$A_1 \times 0 = 0$$

pas d'influence sur le résultat.

Quatrième phase : Décalage de A_1 vers la gauche

$$A_2 = 0101101100$$

Cinquième phase : 3e bit de B = 1

$$A_2 \times 1 = 0101101100$$

à additionner au résultat en R.

$$\begin{array}{r} 01011011 \\ +\ 0101101100 \\ \hline 0111000111\ (455_{10}) \end{array}$$

résultat à replacer en R.

Sixième phase : Décalage de A_2 vers la gauche.

Septième phase : Tous les autres bits de B étant 0, *arrêter l'opération.*

Cette analyse permet de constater les points suivants :

– Dès qu'il n'y a plus que des 0 dans B après avoir examiné un certain nombre de positions, il faut arrêter l'opération. Si B = 0 cet arrêt doit intervenir d'emblée.

– Les décalages successifs de A vont nécessiter de disposer d'un registre de 16 bits que l'on créera, là encore, en disposant de deux registres de 8 bits.

– A chaque opération d'obtention du résultat partiel il faut faire suivre une opération systématique de décalage à gauche supplémentaire de A.

c) Choix de la stratégie de programmation

Celle-ci dépend d'une part de l'algorithme choisi, c'est-à-dire de l'organisation vue au paragraphe précédent, et d'autre part des instructions dont nous disposons.

Pour examiner les bits successifs de B, nous disposons d'une instruction commode qui est le décalage à droite (LSR) où le bit de poids le plus faible est chargé dans le registre de retenue C tandis qu'un 0 vient se loger en bit de poids fort (*cf. fig.* E. 11. 16).

Pour décaler à gauche la valeur A en utilisant deux registres il faut considérer les poids faibles d'une part, les poids forts d'autre part.

Le simple décalage à gauche (ASL) du premier octet introduit un 0 en bit de poids le plus faible et loge le bit de poids le plus fort en C.

Le décalage à gauche de l'octet de poids fort doit reprendre le bit qui est en C et l'introduire à droite. Il faut alors utiliser la rotation à gauche (ROL).

La figure E. 11. 1 montre l'état des deux registres avant les opérations ASL et ROL et l'état de ces deux registres après l'opération. On a bien obtenu un décalage à gauche général, comme si les deux registres n'en formaient qu'un de 16 bits.

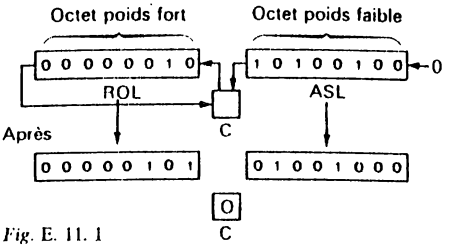


Fig. E. 11. 1

Nous utiliserons alors les registres 9100 et 9101 pour ces décalages à gauche successifs.

Comme, au début, 9101 contient A non décalé, si nous voulons conserver cette donnée, il faudra la loger ailleurs, par exemple en 9105.

De même pour B, qu'il faudra loger en 9106, puisqu'en 9102 nous lui ferons subir des décalages à droite.

d) Programmation

Tout d'abord, faire les rangements dont nous venons de parler :

```
(1)  LDA  0191  (s'habituer à l'inversion des
(2)  STA  0591  adresses)
(3)  LDA  0291
(4)  STA  0691
```

ensuite, par précaution, mettre à zéro les registres 9100, 9103 et 9104

```
(5)  LDA  # 00
(6)  STA  0091
(7)  STA  0391
(8)  STA  0491
```

et vérifier tout de suite que B n'est pas nul

```
(9)  LDA  0291
(10) BNE  XX
(11) (BRK JMP ... RTS)
```

Nous ne connaissons pas encore l'adresse relative XX. Nous compléterons plus tard.

L'instruction BRK entraîne l'arrêt du programme mais il faut la répéter trois fois pour qu'elle agisse. Souvent cette instruction sera remplacée par un Jump à un autre programme (celui de sortie sur écran des résultats par exemple) ou par RTS (Retour au programme principal).

Nous imaginerons ici de mettre plutôt un RTS.

Cette instruction marquera donc la fin du programme si BNE a été ignorée (BNE : Brancher si 9102 n'est pas égal à zéro), donc si B est nul.

– B n'étant pas nul, il faut commencer les opérations de multiplication.

Nous venons de charger 9102 (B) en accumulateur. L'instruction LSR (décalage à droite) peut être tout de suite écrite (elle porte sur le contenu de l'accumulateur).

Le bit le plus à droite de B passe dans le registre C, la nouvelle valeur obtenue doit être rangée à nouveau en 9102. D'où :

```
(12) LSR
13)  STA  0291
```

Puis il nous faut examiner la « retenue ».

BCS (Branch if carry set) nous orientera vers les opérations donnant le premier résultat puis le décalage à gauche de A

```
(14)  BCS  XX
(15)  CLC        (mettre à zéro la retenue)
(16)  LDA  0191
(17)  ADC  0491
(18)  STA  0491
(19)  LDA  0091
(20)  ADC  0391
(21)  STA  0391
```

Ce programme suggère les explications suivantes :

(15) CLC : la retenue, qui a servi au branchement, doit être mise à zéro car nous allons faire une addition.

(16) Chargement de 9101 qui contient non pas forcément A, mais la valeur obtenue par décalages successifs à gauche. Cette valeur est censée occuper les deux registres 9101 et 9100.

(17) L'octet de poids faible de cette valeur est additionné aux résultats intermédiaires (même si pour l'instant ces résultats sont nuls) contenus en 9103 (poids forts) et 9104 (poids faible). Nous commençons par les poids faibles.

(18) Et nous reversons le résultat en 9104.

(19) (20) (21) Même opération pour les octets de poids forts cette fois, mais en prenant soin de ne pas mettre CLC car il faut reporter la retenue éventuelle.

Noter la nécessité de programmer ces opérations de façon suffisamment générale pour qu'elles servent non seulement à la première opération, mais à chacune des opérations successives.

— *Décalage de A*

Le résultat intermédiaire étant enregistré, il faut procéder au décalage à gauche de A par les opérations ASL et ROL déjà étudiées :

```
(22)  LDA  0191
(23)  ASL
(24)  STA  0191
(25)  LDA  0091
(26)  ROL
(27)  STA  0091
```

Tout est maintenant en place pour recommencer avec l'examen du bit suivant de B.

```
(2)  JMP  (9)   (adresse provisoire de l'instruction N° 9)
```

Les instructions suivantes, jusqu'à (14) vont se dérouler. Supposons que cette fois le bit multiplicateur de B soit 0. (14) BCS est ignorée. Il faut alors prévoir un nouveau branchement BCC (Branch if carry clear).

```
(14 bis)  BCC  XX
```

Ce branchement doit court-circuiter l'addition des résultats partiels mais respecter le nouveau décalage à gauche de A, qui est déjà programmé de (22) à (27).

BCC doit donc renvoyer à (22).

e) « Listing » du programme

N° de la mémoire	Mnémoniques	Codes 6502
9000-1-2	LDA 0191	AD 0191
9003-4-5	STA 0591	8D 0591
9006-7-8	LDA 0291	AD 0291
9009-A-B	STA 0691	8D 0691
900C-D	LDA # 00	A9 00
900E-F-10	STA 0091	8D 0091
9011-12-13	STA 0391	8D 0391
9014-15-16	STA 0491	8D 0491
9017-18-19	LDA 0291	AD 0291
901A-1B	BNE (XX)	DO 01
901C	RTS	60
901D	LSR	4A
901E-1F-20	STA 0291	8D 0291
9021-22	BCS (XX)	BO 02
9023-24	BCC (XX)	90 13
9025	CLC	18
9026-27-28	LDA 0191	AD 0191
9029-2A-2B	ADC 0491	6D 0491
902C-2D-2E	STA 0491	8D 0491
902F-30-31	LDA 0091	AD 0091
9032-33-34	ADC 0391	6D 0391
9035-36-37	STA 0391	8D 0391
9038-39-3A	LDA 0191	AD 0191
903B	ASL	OA
903C-3D-3E	STA 0191	8D 0191
903F-40-41	LDA 0091	AD 0091
9042	ROL	2A
9043-44-45	STA 0091	8D 0091
9046-47-48	JMP 1790	4C 1790

Ayant écrit ce programme, il reste à déterminer les adresses relatives des branchements en comptant le nombre d'octets qui séparent l'instruction de branchement de l'adresse à laquelle il faut se brancher. Nous obtenons alors

```
BNE 01
BCS 02
BCC 13H (19 octets)
```

2. Rangement d'une suite de nombres

Cet exercice guidé entraîne à l'utilisation de l'adressage indirect indexé.

Il s'agit de loger les nombres hexadécimaux de 0 à FF dans les mémoires 9000 à 90FF.

Nous utiliserons l'index X à la fois comme registre d'index et comme registre de la valeur à ranger.

Voici la solution proposée :

```
(1)  LDX # 00      : charger la valeur 0 dans X
(2)  TXA           : transférer le contenu de X
                     dans l'accumulateur
(3)  STA 0090, X   : ranger en mémoire 9000
(4)  INX           : incrémenter X
(5)  BNE (2)       : si X n'est pas égal à 0,
(6)  BRK             retourner en (2) sinon arrêter
(7)  BRK
(8)  BRK
```

le programme codé prend alors la forme

```
9100-1   A2   00
9102     8A ←──────┐
9103-4-5 9D   0020 │
9106     E8        │
9107-8   D0   F9 ──┘  on notera la valeur de l'adresse
9109     00           relative
910A     00
910B     00
```

3. Tableau de correspondance hexadécimal-décimal

En s'inspirant de l'exemple précédent, essayez de rédiger un programme permettant de trouver la correspondance entre un nombre hexadécimal et le nombre décimal qui le traduit.

Nous ne pouvons aller que jusqu'à 99 en décimal, donc à 63 en hexadécimal.

Le programme donnera, dans chaque regi d'adresse 9000 à 9063 le nombre décimal correspond aux deux derniers chiffres de l'adresse. Par exempl

en mémoire 900A : le nombre 10 (0001-0000)
en mémoire 9052 : le nombre 82 (1000-0010)

4. Recherche du nombre hexadécimal traduisant nombre décimal donné

Ayant obtenu, grâce au programme précéden tableau de correspondance, il est facile de trouve correspondant décimal d'un nombre hexadécimal.

Il suffit d'appeler l'adresse 90 NN pour avoi correspondance de NN en décimal.

Nous voulons pouvoir faire l'opération invers quoi correspond 43_{10} en hexadécimal par exemple

Imaginez un petit programme de recherche.

INSTRUCTIONS DU 6502 :
Codes hexadécimaux, temps en cycles, nombre d'octets de l'instruction

MNEMONIC	IMPLICITE		ACCUM		ABSOLU		PAGE ZERO		IMMEDIAT		ABS X		ABS Y		IND X		IND Y		Z PAGE X		RELATIF		INDIRECT		Z PAGE Y	
	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N	OP	N
ADC					6D	3	65	3	69	2	7D	3	79	3	61	2	71	2	75	2						
AND					2D	3	25	3	29	2	3D	3	39	3	21	2	31	2	35	2						
ASL			0A	1	0E	3	06	5			1E	7							16	6						
BCC																					90	2				
BCS																					B0	2				
BEQ					2C	4	24	3													F0	2				
BIT																										
BMI																					30	2				
BNE																					D0	2				
BPL																					10	2				
BRK	00	1																								
BVC																					50	2				
BVS																					70	2				
CLC	18	2																								
CLD	D8	2																								
CLI	58	2																								
CLV	B8	2																								
CMP					CD	4	C5	3	C9	2	DD	4	D9	4	C1	6	D1	5	D5	4						
CPX					EC	4	E4	3	E0	2																
CPY					CC	4	C4	3	C0	2																
DEC					CE	6	C6	5			DE	7							D6	6						
DEX	CA	2																								
DEY	88	2																								
EOR					4D	4	45	3	49	2	5D	4	59	4	41	6	51	5	55	4						
INC					EE	6	E6	5			FE	7							F6	6						
INX	E8	2																								
INY	C8	2																								
JMP					4C	3																	6C	5		
JSR					20	6																				
LDA					AD	4	A5	3	A9	2	BD	4	B9	4	A1	6	B1	5	B5	4						
LDX					AE	4	A6	3	A2	2	BE	4							B6	4					B6	4
LDY					AC	4	A4	3	A0	2	BC	4							B4	4						
LSR			4A	1	4E	6	46	5			5E	7							56	6						
NOP	EA	2																								
ORA					0D	4	05	3	09	2	1D	4	19	4	01	6	11	5	15	4						
PHA	48	3																								
PHP	08	3																								
PLA	68	4																								
PLP	28	4																								
ROL			2A	1	2E	6	26	5			3E	7							36	6						
ROR			6A	1	6E	6	66	5			7E	7							76	6						
RTI	40	6																								
RTS	60	6																								
SBC					ED	4	E5	3	E9	2	FD	4	F9	4	E1	6	F1	5	F5	4						
SEC	38	2																								
SED	F8	2																								
SEI	78	2																								
STA					8D	4	85	3			9D	5	99	5	81	6	91	6	95	4						
STX					8E	4	86	3																	96	4
STY					8C	4	84	3																		
TAX	AA	2																								
TSX	BA	2																								
TXS	9A	2																								
TYA	98	2																								

CHAPITRE 12

Les interfaces d'un micro-ordinateur

INTRODUCTION

Si l'on ne dispose que d'un ensemble composé d'un clavier, d'un microprocesseur et d'une mémoire centrale on ne peut rien faire, ou plutôt les opérations ne pourront se faire qu'à l'aveuglette, et rien ne nous permettra de prendre connaissance des résultats.

Il est donc indispensable de disposer pour le moins d'un système de visualisation (afficheurs « 7 segments », écran de télévision ou imprimante).

On constate alors vite qu'il est très difficile de tirer un bon parti de l'ensemble si, chaque fois que l'on coupe le courant d'alimentation, tout le travail de programmation réalisé disparaît des mémoires volatiles. D'où la nécessité de « sauver » ces programmes sur bande magnétique ou sur disquette, puis de les recharger à la demande en mémoire centrale.

Tous ces organes (écran, imprimante, lecteur à cassette, disquette...) sont appelés des périphériques au même titre que le clavier. C'est eux qui forment, avec le microprocesseur et sa mémoire l'ensemble appelé micro-ordinateur.

Pour gérer les échanges avec ses périphériques, le microprocesseur disposera de circuits intermédiaires que l'on appelle des interfaces.

Nous allons en étudier quelques-uns des plus courants.

1. PORTS D'ENTRÉE/SORTIE EN PARALLÈLE

Un certain nombre d'interfaces sont destinées à envoyer ou à recevoir des groupes de 8 bits en parallèle. Ces interfaces ont reçu le nom de PIO (Programmable Input Output Device), PIA (Peripheral Interface Adapteo), PPI (Programmable Peripheral Interface).

Nous allons étudier, dans son principe, la structure et le fonctionnement de ce type d'interface.

1.1 Structure d'un PIA

Le PIA se présente sous forme d'un circuit intégré à 40 broches (voir en *fig.* 12. 1 le brochage du PIA 6520).

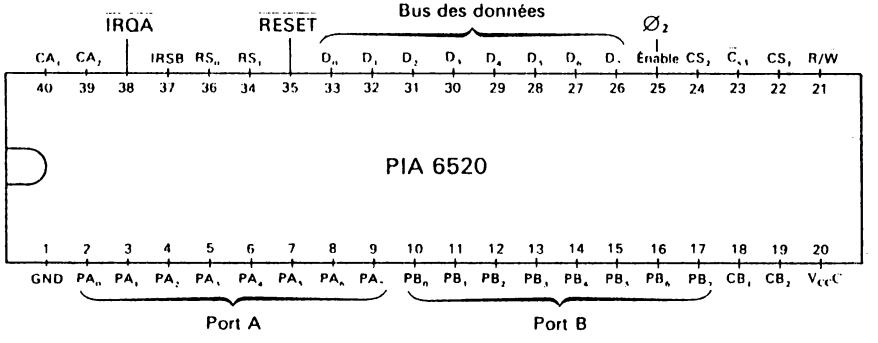


Fig. 12. 1

Huit broches serviront à recevoir les données venant du microprocesseur ou à les transmettre à celui-ci (bornes D_0 à D_8).

Vers l'extérieur deux groupes de huit broches peuvent être reliés à deux périphériques différents et constituent ce que l'on appelle les *ports* d'entrée/sortie (P_A pour le port A et P_B pour le port B).

En effet, tout comme les ports sont destinés à charger ou décharger des marchandises, chacun de ces ports d'entrée/sortie pourra, à la demande, recevoir ou envoyer des informations.

Bien mieux, pour chaque port, c'est chaque borne qui peut être «programmée» pour être réceptrice ou expéditrice.

Pour cette programmation chaque port dispose d'un *registre de direction*, registre à huit cellules permettant de programmer chaque borne d'entrée/sortie en entrée ou en sortie par l'intermédiaire d'un jeu de portes «3 états».

On voit, sur la figure 12. 2, que si l'on met le registre S à 0, la circulation des informations se fera de P_A vers D (réception d'informations) alors que si S est à 1, le sens est celui de l'expédition d'informations de D vers le port P_A.

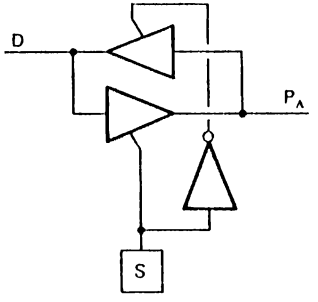


Fig. 12. 2

Outre ces deux registres de direction, le PIA comporte aussi deux registres tampons, R_A et R_B, un pour chaque port, permettant de stocker l'information entre le microprocesseur et les périphériques.

Enfin chaque port est doté d'un troisième registre qui est le *registre de contrôle*. Ce registre régit les commandes internes du PIA suivant les ordres reçus de l'extérieur (périphériques) ou du microprocesseur.

1.2 Fonctionnement du PIA

Vis à vis du microprocesseur le PIA doit apparaître comme une mémoire RAM. En effet nous avons vu que les actions possibles d'un microprocesseur vers l'extérieur étaient la lecture ou l'écriture de données à la suite d'un adressage.

Le PIA doit donc pouvoir être adressé et mis en lecture (réception de données) ou mis en écriture (envoi de données).

1.2.1 L'adressage du PIA

Comme il y a déjà un bus de données (D_0 à D_7) et deux ports d'entrée/sortie (P_{A_0} à P_{A_7} et P_{B_0} à P_{B_7}) sur le PIA, ceci occupe 24 broches sur les 40 disponibles. Un adressage sur 16 broches saturerait toutes les disponibilités. Or il faut encore l'alimentation, les commandes de lecture/écriture et quelques autres commandes que nous allons étudier.

En fait il n'est pas nécessaire d'avoir 16 fils d'adresse sur le PIA.

En effet, en règle très générale, dans tout micro-ordinateur, un groupe d'adresses est réservé aux périphériques. Leur désignation dépendra du constructeur. Parmi ce groupe d'adresses, un sous-groupe va être réservé au PIA.

Ce sera par exemple le sous-groupe des adresses 8000, 8001, 8002 et 8003. Toutes ces adresses ont en commun les mêmes valeurs des bits d'adresse de rang A_2 à A_{15}. Il suffira donc qu'un décodeur, intercalé entre le microprocesseur et le PIA, reçoivent les fils A_2 à A_{15} du bus d'adresse et envoie un niveau haut sur une borne CS (Chip Select) pour que le PIA soit activé. Il restera deux fils d'adresse à brancher directement sur le PIA : A_0 et A_1 qui seront reliés aux bornes RS_0 et RS_1 (cf. *fig.* 12. 3).

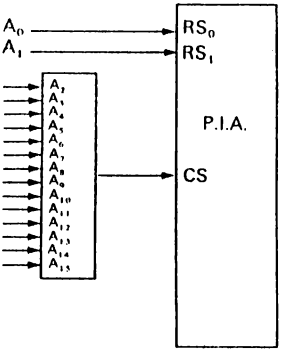


Fig. 12. 3

Sur le PIA classique de Motorola, la borne CS est en fait remplacée par 3 bornes (CS_1, CS_2 et $\overline{CS_3}$) ce qui permet de mettre en parallèle jusqu'à huit PIA par le jeu de sélections différentes. Outre

le fait qu'il est rare d'avoir à disposer autant d'interfaces sur un microprocesseur, il en résulte qu'il ne reste plus assez de bornes pour adresser séparément chacun des 6 registres internes du PIA (registre de contrôle, registre de direction et buffer de chaque port).

On ne dispose donc que de 4 adresses (sur RS_0 et RS_1) pour adresser 6 registres. Ceci n'est possible que par un artifice :

– l'adresse 01 sélectionne le registre de contrôle A,

– l'adresse 11 sélectionne le registre de contrôle B.

On peut écrire dans chacun de ces registres.

Or, si le bit deux (3e bit) d'un registre de contrôle est à 0, on « ouvre la communication » avec le registre de direction aux adresses 00 pour le côté A et à 10 pour le côté B.

Si ce bit deux est à 1, c'est le buffer du port A qui est sélectionné par l'adresse 00, ou le buffer du port B par l'adresse 10.

Cette disposition complique quelque peu la programmation du PIA.

1.2.2 Programmation du PIA

Notons, avant d'aborder cette programmation, que le PIA est muni d'une borne « RESET » qui est reliée au RESET général. A la mise sous tension tous les registres du PIA sont donc mis à 0 ce qui a pour conséquence :

a) que les ports sont programmés en récepteurs d'information. Ainsi il ne peut pas y avoir de créneau intempestif de tension envoyé du PIA vers les périphériques,

b) que les registres adressés par 00 ou 10 seront d'emblée les registres de direction et non pas les buffers d'entrée/sortie.

Si le bloc d'adresses du PIA est 8000 à 8003 (en hexadécimal) et si l'on veut organiser le port A de telle sorte que les bornes A_0 à A_3 soient réceptrices et les bornes A_4 à A_7 soient expéditrices, le programme sera le suivant (en utilisant les mnémoniques du 6502).

```
LDA # F0  (1 1 1 1 0 0 0 0)
STA 8000  adresse du registre de direction, le
          bit 2 du registre de contrôle ayant
          été mis à 0 par RESET)
LDA # 04  (0 0 0 0 0 1 0 0)
STA 8001  (adresse du registre de contrôle. Le
          bit 2 est maintenant à 1).
LDA MEM   (adresse d'une mémoire)
STA 8000  (adresse du buffer du port A : les
          4 bits de poids forts sont expédiés
          vers le périphérique).
```

Si l'on veut lire les valeurs envoyées par le périphérique, on écrira :

```
LDA 8000
STA MEM (éventuellement).
```

On voit que les instructions du PIA sont les mêmes que celles qui s'adressent à une mémoire. Il y a en effet une borne R/$\overline{W}$ sur le PIA. La lecture (R) correspondra à la réception d'une information et l'écriture ($\overline{W}$) à l'envoi d'une information.

1.2.3 Contrôle des échanges d'informations

Un bon fonctionnement de cet interface exige que le périphérique puisse informer le microprocesseur

– que les informations qui lui sont destinées sont bien arrivées,

– que le microprocesseur ne charge pas plusieurs fois une même information.

On trouve ici le même problème que celui rencontré dans la gestion d'un clavier. La solution est du même type : la création de drapeaux.

Pour chaque port A et B le périphérique est relié au PIA par deux fils de « contrôle » C1 et C2 qui permettent l'échange des signaux correspondant à ces drapeaux.

De son côté le microprocesseur est informé tout d'abord par la liaison IRQ (« Interrupt Request » ou demande d'interruption). Son action sera étudiée au § 3 de ce chapitre.

Par ailleurs le microprocesseur peut écrire dans le registre de contrôle, notamment pour abaisser le drapeau IRQ, répondant en quelque sorte au périphérique qui demande la parole. Ce dialogue par des drapeaux alternativement levés et abaissés est désigné par le terme anglais de « handshaking » (ou « poignée de main »).

On trouvera, en annexe 1, le rôle des registres de contrôle d'un PIA.

2. INTERFACE PARALLÈLE/SÉRIE

On peut songer à utiliser un PIA pour sauver un programme que l'on vient d'écrire sur un support de type bande magnétique. Chaque octet lu par le microprocesseur serait envoyé sur le PIA pour être lu à son tour par le périphérique. Il serait alors nécessaire de disposer d'un enregistreur du type magnétophone ayant 8 têtes de lecture, en parallèle. Ces appareils sont très coûteux et on les réserve plutôt à la prise de son en studio.

Utiliser un petit magnétophone à cassette rudimentaire paraît être une meilleure solution, même si l'on y perd un peu de temps. C'est possible, à condition de transmettre en série les octets chargés en parallèle. D'où la création d'interfaces parallèle/série.

2.1 Analyse du problème

On peut imaginer que le microprocesseur charge chaque mot du programme, puis l'ayant chargé, applique 8 fois l'instruction « rotation à droite » pour envoyer chaque bit recueilli sur la tête d'écriture du magnétophone.

Le rythme de transfert pourrait être alors d'un bit toutes les 12 µs, ce qui correspond à une fréquence de plus de 80 kHz. Notons tout de suite que cette fréquence est bien trop élevée pour un simple magnétophone qui parvient à peine à enregistrer du 15 kHz.

De plus, comment sera-t-il possible de reconnaître le poids de chaque bit, celui qui commence un mot ou celui qui le termine? Il est à nouveau fort coûteux de synchroniser un magnétophone sur le microprocesseur et on préfèrera considérer que le magnétophone est un organe *asynchrone* par rapport au microprocesseur.

Les constructeurs se sont donc attachés à concevoir un interface capable de s'adapter au rythme et aux caprices du périphérique le moins élaboré qui soit : le petit dictaphone portatif.

Ce circuit se trouve dans les catalogues sous différentes dénominations telles que USART (Universal Synchronous Asynchronous Receiver Transceiver), UART ou encore ACIA (Asynchronous Communications Interface Adapter).

2.2 Structure d'un ACIA

S'agissant de transmettre à un périphérique des séries de bits (ou de les recevoir de lui), les bornes de sorties seront donc limitées à deux fils : l'un pour la sortie vers le périphérique (T × Data), l'autre pour l'entrée des informations venant du périphérique (R × Data) (cf. *fig.* 12. 4).

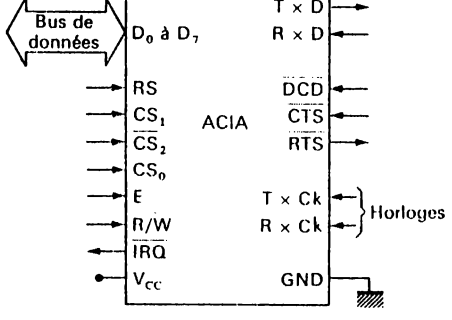


Fig. 12. 4

Du côté microprocesseur on trouvera les huit bornes correspondant au bus de données, une borne d'adresse RS permettant d'adresser deux registres différents et trois bornes CS (Chip Select) comme pour le PIA. En outre on trouvera les bornes R/$\overline{W}$, « Enable » ainsi que l'alimentation.

Trois bornes appelées $\overline{DCD}$, $\overline{CTS}$ et $\overline{RTS}$ sont destinées à la connexion avec un « modem » (voir annexe).

Enfin deux bornes T × Ck et R × Ck correspondent à des entrées d'horloge dont nous allons voir l'utilité.

Du côté microprocesseur, les données transmises par le bus des données aboutissent dans un registre tampon appelé TDR (Transmitter Data Register).

On imagine assez bien que la sortie série se fera grâce à un registre à décalage commandé par une horloge. Ce registre est distinct du TDR. On l'appelle TSR (Transmitter Shift Register). L'horloge provoquant le décalage est fournie par l'entrée T × Ck et la sortie des données en série s'effectue par la borne T × Data (ou T × D).

2.3 Fonctionnement d'un ACIA

Pour bien comprendre le fonctionnement d'un ACIA il suffit de noter les principes qui ont guidé sa conception.

a) L'interface doit envoyer des éléments binaires en série au périphérique à la fréquence à laquelle celui-ci est capable de les recevoir.

Il faudra donc faire attendre le microprocesseur après chaque transmission d'octet, le périphérique étant beaucoup plus lent que le microprocesseur.

b) La fréquence de réception par l'interface des données restituées par le périphérique sera pratiquement la même : c'est le même magnétophone à cassette qui sert pour emmagasiner ou restituer des données, et il doit tourner à la même vitesse, aux petites variations près dues aux irrégularités du moteur ou de son alimentation.

c) Il sera nécessaire de distinguer chaque octet l'un de l'autre et d'être averti des erreurs possibles de transmission.

2.3.1 Données transmises par le microprocesseur

– L'ACIA étant adressé, un premier octet peut lui être transmis qui va être mémorisé dans TDR.

– De TDR les données sont reportées dans TSR.

– L'horloge de transmission (externe à l'ACIA) va mettre en sortie les bits successifs de

l'octet à transmettre, au rythme acceptable par le périphérique.

Mais cette solution implique quelques données ou actions complémentaires pour indiquer le début et la fin d'un caractère (octet).

Il faut tout d'abord une position de départ bien précise. Celle-ci est obtenue par une commande interne de « Reset » qui inhibe les signaux d'horloge et met un compteur interne à 0. Un signal haut parvient sur la tête d'enregistrement du magnétophone par la borne T × D.

C'est à ce moment-là que TDR peut être rempli, pour être vidé dans TSR. La transmission est prête à être effectuée.

Un signal bas est établi sur T × D. Ce signal met en route le compteur interne qui passe à 1.

Le coup d'horloge suivant provoque le premier décalage : le premier bit de donnée sort de TSR pour s'inscrire sur la bande magnétique.

Au neuvième coup d'horloge les 8 bits auront été transmis. Le dixième coup d'horloge provoquera alors un nouveau « reset » qui ramènera à l'état de départ avec un niveau haut sur T × D. Un nouveau cycle peut être réalisé.

2.3.2 *Données en provenance du périphérique*

De même que TSR peut être chargé en parallèle à partir de TDR et déchargé en série sur T × D, de même un autre registre, appelé RSR peut être chargé en série à partir de R × D et déchargé en parallèle dans un registre RDR.

L'ACIA est donc composé de deux parties assez symétriques, où les courants de circulation d'informations sont inversés (*fig.* 12. 5).

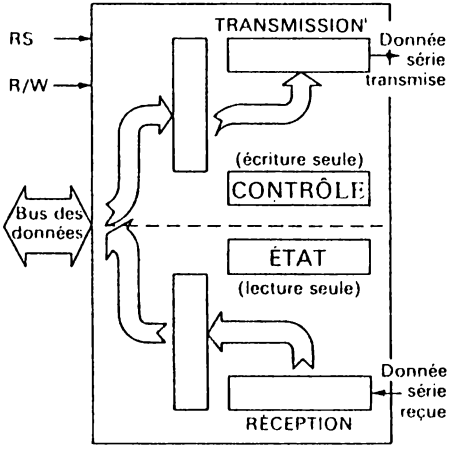


Fig. 12. 5

Analysons ce qui se passe lorsque c'est la partie réceptrice (R comme Receiver) qui est en fonctionnement.

Le magnétophone dont la bande magnétique contient un enregistrement (souvent il s'agit d'un programme qui a été « sauvé ») est mis en route tandis que le microprocesseur est aiguillé sur un programme de lecture de l'ACIA.

Avant l'envoi du premier caractère un niveau haut est appliqué sur R × D. Dès qu'un niveau bas apparaît une horloge R × Ck entre en action.

Cette horloge a une fréquence de 16 fois la fréquence de l'horloge T × Ck. Elle incrémente un compteur qui, par le biais d'un réseau logique, va déclencher un échantillonnage de la tension sur R × D au bit n° 8 de l'horloge (cf. *fig.* 12. 6).

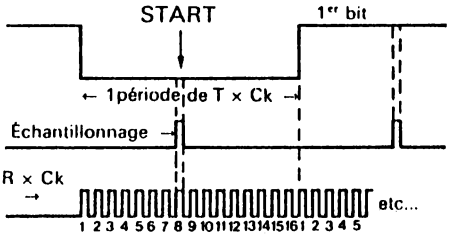


Fig. 12. 6

Le premier échantillonnage donnera le signal « start », les huit suivants seront accompagnés de l'introduction des valeurs ainsi échantillonnées dans le registre RSR. Le dixième échantillonnage doit détecter la valeur 1 qui inhibe l'horloge, remet le compteur à 0 et déclenche le transfert de RSR dans RDR.

On voit l'avantage de ce système d'échantillonnage. Comme il a lieu au milieu du créneau de tension correspondant à la valeur du bit, il y a peu de risques qu'une variation de vitesse du moteur du magnétophone (due par exemple à de légères variations de l'alimentation) ou un glissement de la bande magnétique, ou même des variations de la longueur de cette bande dues à un léger étirage ou aux conditions hygrométriques puissent amener un décalage tel que l'échantillonnage ait lieu en dehors de la zone présumée du bit à lire.

Ceci est d'autant moins possible que ces variations ne se cumulent que sur les huit bits d'un octet et ne concerneront plus l'octet suivant qui fera l'objet d'un nouveau « start ».

On mesure ici l'intérêt que peut avoir le bit de parité dans le transfert d'un caractère. Si un mauvais fonctionnement du magnétophone entraîne de fausses lectures (signe que l'appareil est à réviser ou qu'il faut changer ses piles) le bit de parité permettra très vite de détecter l'erreur.

2.3.3 Registre de contrôle

L'ACIA peut fonctionner sous différents modes :

a) En émission ou en réception.
b) Avec des mots de sept ou huit bits.
c) Avec un ou deux bits de stop.
d) Avec un bit de parité ou sans bit de parité.
e) Le bit de parité étant à 1 pour un nombre pair de un ou pour un nombre impair.
f) Avec une horloge d'échantillonnage à 16 fois ou à 64 fois la fréquence de T × Ck.

Il s'agit là d'une « carte » où l'on peut choisir son menu.

Ce choix est fait par inscription de 0 ou de 1 dans un registre dit de contrôle qui, une fois rempli, définit les règles de fonctionnement de l'ACIA.

Seul le microprocesseur peut avoir accès à ce registre, en écriture.

Pour savoir comment composer ce menu il suffira de lire la documentation fournie par chaque constructeur.

2.3.4 Registre d'états

La gestion interne de l'ACIA provoque la mise en place de différents drapeaux qui sont inscrits dans un registre d'états. Ce registre ne peut être écrit que par l'ACIA. Le microprocesseur ne peut que le lire.

Ces drapeaux indiquent si le registre de transmission est vide (le microprocesseur peut alors y écrire une nouvelle donnée), si le registre de réception est plein (le microprocesseur peut y lire la nouvelle information); ils donnent également des informations sur la qualité de la réception :

– erreur de format, le mot reçu n'a pas le format correspondant à celui indiqué dans le registre de contrôle,
– absence de bit de stop alors que le registre de contrôle le prévoit,
– erreur de parité,
– recouvrement de deux mots, le microprocesseur n'ayant pas lu le précédent.

Deux autres bits sont spécifiques de l'utilisation d'un modem (voir annexe).

Le bit d'état le plus à droite est relatif aux interruptions que nous étudierons un peu plus loin.

2.3.5 Adressage de l'ACIA

Le décodage de la partie « poids forts » de l'adresse étant réalisé, il servira à sélectionner le circuit grâce aux bornes CS. Il ne reste plus qu'un fil de sélection, donc deux adresses possibles : × × × 0 ou × × × 1. Or il y a quatre registres à adresser : le registre TDR, le registre SDR, le registre de contrôle et le registre d'état.

Ces registres peuvent être groupés en deux catégories distinctes :

– ceux qui ne peuvent être que lus par le microprocesseur : SDR et le registre d'état,
– ceux où le microprocesseur ne peut qu'écrire : TDR et le registre de contrôle.

C'est donc la commande R/W qui constituera le deuxième bit d'adressage :

L'adresse 0 désignant soit le registre de contrôle soit le registre d'état, et l'adresse de l'ACIA étant par exemple 9100, l'instruction LDA 9100 atteindra le registre d'état (lecture) et l'instruction STA 9100 atteindra le registre de contrôle (écriture).

A son tour l'adresse 1 atteindra TDR par l'instruction STA 9101 et SDR par LDA 9101.

3. LES INTERRUPTIONS

Tout microprocesseur possède des broches d'interruption par lesquelles il est relié à ses périphériques. Nous avons évoqué la ligne IRQ (demande d'interruption) à propos des interfaces précédentes. Outre IRQ, on trouve aussi l'interruption NMI (interruption non masquable) et le RESET. Pour qu'une de ces interruptions soit active il faut que le signal qui y correspond passe à 0, c'est pourquoi les bornes correspondantes sont notées $\overline{IRQ}$, $\overline{NMI}$ et $\overline{RESET}$. Pour plus de facilité nous ne mettrons pas les barres dans les paragraphes qui suivent.

3.1 Le « Reset »

Nous avons déjà vu l'utilité du « Reset » à plusieurs reprises : c'est une interruption qui est en général utilisée au moment du démarrage du système. A la mise sous tension d'un micro-ordinateur l'état des registres peut être quelconque et il est utile de les mettre à 0 (ou quelquefois à 1), bref de les initialiser pour permettre un démarrage correct.

Le « Reset » applique une tension nulle sur la borne correspondante pendant quelques millisecondes. Après quoi le microprocesseur sera orienté vers un programme de démarrage.

3.2 L'interruption « IRQ »

En général cette interruption provient de l'interface de l'un des périphériques (PIA, ACIA, clavier...) et signale au microprocesseur que ce périphérique est prêt à lui transmettre une information.

Lorsque cette interruption se présente le microprocesseur est le plus souvent en train de travailler (on peut considérer que dès qu'il est sous

tension il ne se repose plus) sur un programme interne ou sur un programme qui lui a été transmis (qui figure en RAM). Appelons ce programme « programme principal », ou, en abrégé, P.Pr).

L'interruption invite l'unité centrale à interrompre ce programme pour répondre au périphérique.

Cette réponse se fera par saut au programme de réponse au périphérique. Dès lors deux problèmes se posent :

— Comment abandonner P.Pr tout en préservant les possibilités de le reprendre un peu plus haut?

— A quelle adresse de programme sauter pour répondre au périphérique?

3.2.1 Sauvegarde de P.Pr

IRQ apparaissant, il n'est pas question que l'instruction de P.Pr en cours soit interrompue.

Lorsque celle-ci est terminée, l'unité centrale est orientée vers un programme de sauvetage qui doit permettre la reprise ultérieure de P.Pr.

Ce programme* permet de ranger le contenu du compteur de programme, des registres X et Y, de l'accumulateur, du registre d'état, ceci se faisant dans un ordre bien déterminé, les casiers de rangement étant la pile du micro-ordinateur.

Le retour au programme principal se fera ensuite par déstockage de cette pile, en faisant appel au contenu du « pointeur de pile » SP (Stack pointer) qui indique l'adresse du sommet de la pile.

Dès que ce rangement est fait, l'unité centrale peut aborder le saut au programme de réponse du périphérique.

* On notera que ce programme fait parfois partie de la liste des micro-programmes contenus dans l'unité de commande du microprocesseur. Il n'a donc pas besoin d'être écrit par le programmeur et il se déclenche automatiquement. Quelquefois il s'agit d'un programme en ROM.

3.2.2 Saut au programme concernant le périphérique

Chaque micro-ordinateur utilise en fait une solution spécifique et pour la sauvegarde des registres et pour le saut au sous-programme, l'un faisant plus appel au hardware qu'au software, l'autre, au contraire, s'appuyant sur le software.

Si l'on prend l'exemple du 6502, il faut retenir que l'interruption est interprétée comme un JSR (saut à un sous-programme) avec sauvegarde du compteur de programme dans la pile et recherche de l'adresse de début de sous-programme dans les mémoires FFFE et FFFF. C'est en quelque sorte un JSR indirect.

Mais, ni l'accumulateur, ni les registres X et Y ne sont sauvés. C'est donc au programmateur à prévoir, dès le début du sous-programme, la sauvegarde de ces registres si cela s'avère nécessaire pour la reprise ultérieure du programme.

L'adresse contenue en FFFE et F correspondant à des mémoires RAM, on pourra composer à son gré ce sous-programme ou même utiliser les premiers octets à une instruction JMP (saut inconditionnel) renvoyant à une routine située ailleurs.

3.2.3 Masquage de l'interruption IRQ

Dans le registre d'état du microprocesseur une cellule, nommée I comme interruption, peut contenir un masque d'interruption sous la forme d'un 1.

Si I est à 0 l'interruption est prise en compte comme nous venons de le voir. Si I est à 1, l'interruption ne doit pas être prise en compte.

Dès que le microprocesseur a pris en compte une interruption, il met à 1 le registre I de façon à ne pas être interrompu une nouvelle fois jusqu'à la fin de son programme d'interruption.

Si l'on reprend l'exemple du 6502, IRQ survenant, ce qui suppose que I est à 0, le saut au sous-programme s'effectue, les registres sont sauvegardés dans la pile et notamment le registre d'état avec I = 0. Ce n'est qu'à ce moment-là que la valeur 1 est mise dans le registre I.

A la fin du sous-programme les anciennes valeurs des registres sont reprises de la pile et reversées dans les registres, de sorte que I est de nouveau à 0 et une nouvelle IRQ pourra être prise en compte.

3.3 L'interruption NMI

Comme son nom l'indique (Non Masquable Interrupt) l'interruption NMI ne peut pas recevoir de masque.

Par exemple, lorsque l'on frappe sur le clavier d'entrée des données, le microprocesseur doit prendre en compte chaque caractère avant l'arrivée du suivant.

Lorsque le microprocesseur traite un sous-programme NMI, le drapeau I est mis à 1 pour éviter toute interruption IRQ pendant ce temps-là.

3.4 Le « break »

Sur le 6502 en particulier il existe une interruption programmable sous forme d'une instruction particulière déjà vue, l'instruction BRK.

L'exécution de cette instruction entraine la sauvegarde dans la pile du compteur de programme, puis, tout comme pour IRQ, le saut au sous-programme dont l'adresse de début figure en FFFE et FFFF.

De plus un registre spécial d'état, appelé registre B, est porté à la valeur 1.

D'autres micro-ordinateurs ont des instructions de break qui branchent sur des sous-programmes différents de celui de IRQ. Il est bon, chaque fois, de se référer à la documentation du constructeur.

CHAPITRE 12. ANNEXE 1

Rôle des registres de contrôle d'un PIA

Nous avons vu comment chaque port d'un PIA et chaque élément de ce port pouvait être programmé en entrée ou en sortie par la simple mise en mémoire d'un 0 ou d'un 1.

L'idée qui préside à cette sélection est aussi appliquée dans la gestion interne du PIA : il s'agit de réaliser des circuits à aiguillages commandés par des portes « trois-états » ou des portes AND ou NAND, et de se servir de cellules de mémoires pour créer les ordres d'aiguillages.

Pour adapter telle ou telle configuration du circuit électronique, il n'y aura plus qu'à « programmer » les cellules de mémoire. C'est l'exemple le plus frappant de jonction du « software » avec le « hardware ».

Nous allons préciser ici le rôle de chaque bit du mot de contrôle figurant dans le registre de contrôle A d'un PIA. S'agissant du port B, le rôle du registre de contrôle B est très similaire.

Rappelons que ce registre, que nous appellerons CRA, contient 8 bits et qu'il est chargé d'assurer les échanges entre les signaux sur CA1, CA2 et IRQA ainsi que l'aiguillage de l'adressage « registre de direction/port A ». On dit qu'il établit le *protocole* de gestion du port A.

Le rôle des 8 bits est de façon très générale le suivant :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
IRQA1	IRQA2	Commandes CA2			Registres de direction ou port	Commandes CA1	

Détaillons le rôle de chaque bit.

Bit 0 : $B_0 = 0$ les interruptions sur CA1 sont masquées (rappelons que CA1 ne fonctionne qu'en entrée).
$B_0 = 1$ les interruptions sur CA1 sont validées.

Bit 1 : $B_1 = 0$ IRQA passe à 0 sur un front descendant de CA1.
$B_1 = 1$ $\overline{\text{IRQA}}$ passe à 0 sur un front montant sur CA1.

Bit 2 : $B_2 = 0$ sélection du registre de direction
$B_2 = 1$ sélection du port d'entrées/sorties A.

Bits 3, 4 et 5 : ils organisent les commandes de CA2 qui est une ligne bidirectionnelle. Si le bit 5 est à 0, CA2 est programmée en entrée; s'il est à 1, CA2 est programmée en sortie.

Pour chacun de ces cas on a les possibilités de configurations suivantes :

B_5	B_4	B_3	Ligne CA2 programmée en entrée
0	×	0	— Les interruptions venant de CA2 sont masquées.
0	×	1	— Les interruptions venant de CA2 sont validées suivant deux modes possibles :
0	0	1	— Interruptions sur front négatif du signal CA2.
0	1	1	— Interruption sur front positif du signal CA2.

B_5	B_4	B_3	Ligne CA2 programmée en sortie
1	0	0	– CA2 passe à 1 en cas d'interruption sur CA1 puis repasse à 0 à la lecture du port A.
1	0	1	– La lecture du port A met CA2 à 0 pendant un cycle d'horloge.
1	1	0	– Mise à 0 de CA2.
1	1	1	– Mise à 1 de CA2.

Bit 6 : c'est un indicateur d'interruptions sur CA2. Il passe à 1 lorsque une interruption est activée sur CA2. Il est mis à 0 à la lecture du port A.

Bit 7 : même rôle que le bit 6 mais concernant CA1.

CHAPITRE 12. ANNEXE 2

Liaison d'un ACIA avec un modem

Un MODEM (Modulateur-Démodulateur) est le circuit chargé de coder ou de décoder des informations binaires pour leur transmission sur des lignes téléphoniques « analogiques ».

En attendant la « digitalisation » complète des lignes téléphoniques qui est encore loin d'atteindre les lignes des particuliers, le MODEM est très utile pour l'échange d'informations entre ordinateurs situés à quelque distance l'un de l'autre ou entre un ordinateur et un terminal.

Le système de codage du digital en analogique consiste ici à adopter deux fréquences différentes pour le 0 et pour le 1. Deux codages différents sont utilisés :

a) 0 transformé en une fréquence de 1 070 Hz
1 transformé en une fréquence de 1 270 Hz
b) 0 transformé en une fréquence de 2 025 Hz
1 transformé en une fréquence de 2 225 Hz

Le MODEM contrôle la transmission en établissant un « dialogue » avec le MODEM situé à l'autre extrémité de la ligne de transmission et qui joue un rôle symétrique du sien.

Utilisé avec l'ACIA, le MODEM permettra à l'utilisateur de faire donc parvenir, via la ligne téléphonique, des informations à une installation dotée elle-même d'un MODEM et d'un ACIA (*fig.* 12. 7).

Sur l'ACIA, certaines broches et certains bits du registre de contrôle sont réservés au MODEM. Ce sont les bornes $\overline{CTS}$, $\overline{RTS}$, $\overline{DCD}$ et les bits 5 et 6 du registre de contrôle.

Enfin les bits 2 et 3 du registre d'état ont trait aux échanges ACIA-MODEM (*fig.* 12. 8).

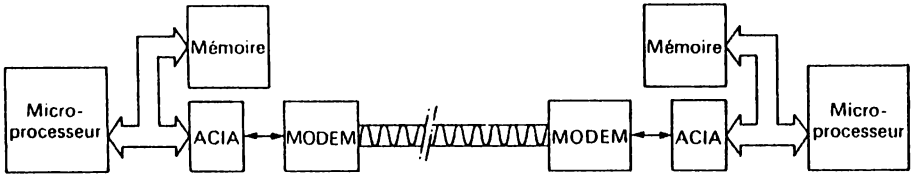


Fig. 12. 7

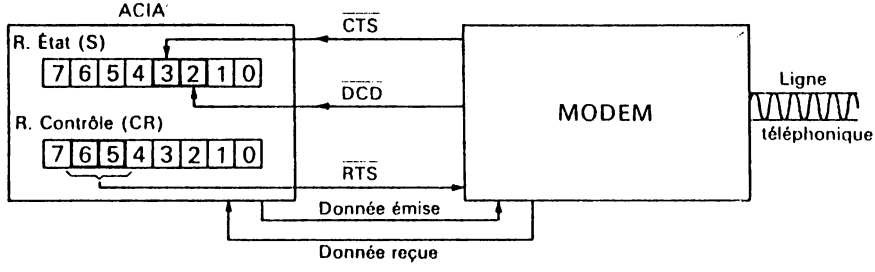


Fig. 12. 8

Le signal $\overline{RTS}$ issu de l'ACIA permet au microprocesseur de commander le MODEM en lui imposant de se préparer à émettre un mot. Ceci s'obtient en donnant à CR_6 (bit 6 du registre de contrôle) la valeur 0 ou en mettant CR_5 et CR_6 à 1.

Ayant reçu ce signal, le MODEM répond par $\overline{CTS}$ («prêt à envoyer!»). $\overline{CTS}$ agit sur le bit d'état relatif au registre de transmission (S_3).

Si le MODEM n'est pas prêt, $\overline{CTS} = 1$ et $S_3 = 0$.

Le signal $\overline{DCD}$ est utilisé dans le cas d réception d'informations. Avec $\overline{DCD} =$ MODEM indique qu'il reçoit une onde port de données, le bit d'état S_2 passe à 1 pour indi que le registre de réception est garni.

Lorsque le MODEM ne reçoit plus d'info tion, $\overline{DCD}$ passe à 1. Il peut parfois s'agir d perte accidentelle de la porteuse; dans ce cas action spécifique du microprocesseur doit envisagée.

CHAPITRE 13

Les automates programmables

INTRODUCTION

Nous avons évoqué au cours des chapitres précédents les nombreux domaines dans lesquels le microprocesseur introduit de profonds bouleversements technologiques : le domaine du son tout d'abord, avec l'apparition des disques à enregistrement digital; celui de l'image de télévision où l'on peut prévoir que les expériences actuelles de codification numérique du signal déboucheront bientôt; le domaine des télécommunications, plusieurs fois évoqué, et enfin celui des automatismes.

Aborder chacune des ces techniques dépasserait le cadre imparti à cet ouvrage. Nous nous contenterons de traiter du dernier cas, celui des automatismes, car il est à lui seul très représentatif des simplifications et des progrès que permet le microprocesseur dans un très vaste éventail d'applications allant du jouet d'enfant aux satellites en passant par la machine à laver domestique, l'automobile, les machines-outils ou les installations de l'industrie pétrolière, chimique, nucléaire etc...

1. ARCHITECTURE D'UN AUTOMATE PROGRAMMABLE

La structure générale d'un système de commande basé sur l'emploi d'un automate programmable est présentée en figure 13. 1.

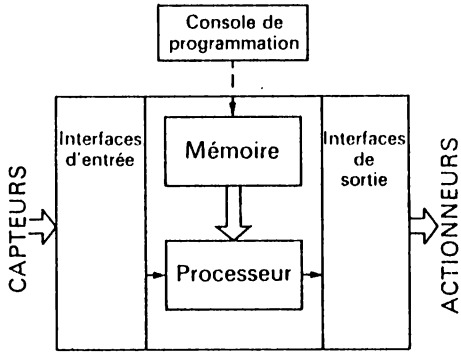


Fig. 13. 1

On y distingue quatre sous-ensembles :

a) les entrées,

b) les sorties,

c) la mémoire où sont enregistrées les instructions du programme de commande ainsi que des données en cours de traitement,

d) le processeur qui exécute les instructions (UCT).

La console de programmation est un accessoire, souvent amovible, utilisé par le concepteur de l'automation pour introduire le programme en mémoire ou mettre au point ce programme.

La figure 13. 2 montre une structure plus détaillée d'un automate programmable relativement performant. Hormis le fait que cet ensemble est largement doté de ce que l'on appelle des « coupleurs » d'entrées ou de sorties, qui sont ce que l'on a appelé précédemment des interfaces entre le microprocesseur et l'extérieur, le schéma est très voisin de celui d'un mini-ordinateur.

Le processeur lui-même peut être un microprocesseur standard capable de réaliser les opérations logiques et arithmétiques classiques, ainsi que les instructions ayant trait au déroulement du programme : sauts conditionnels et inconditionnels, traitement de boucles (routines) etc...

La mémoire contiendra le programme qui, s'il doit résider en permanence, aura la particularité d'être en EPROM, les RAM étant utilisées pour stocker des données. Quant aux ROM elles contiendront le moniteur, programme qui assure la gestion du système interne et notamment le démarrage et la gestion des interruptions (on peut penser que, dans ce domaine, ce dernier programme doit prévoir de nombreux cas).

L'assembleur est le programme qui permet de programmer en utilisant des mnémoniques. Un automate aura un assembleur original, orienté vers les applications auxquelles il est destiné, et permettant d'utiliser des mnémoniques très simples pour les instructions qui y correspondent. Nous en verrons quelques exemples plus loin.

Le nombre des interfaces (coupleurs) est lié à la variété des informations à recevoir ou à transmettre. C'est là que l'automate diffère du micro-ordinateur. Nous allons voir tout d'abord comment recueillir ces informations par des «capteurs», puis comment transformer les informations envoyées par l'automate en commandes physiques, grâce à des «actionneurs».

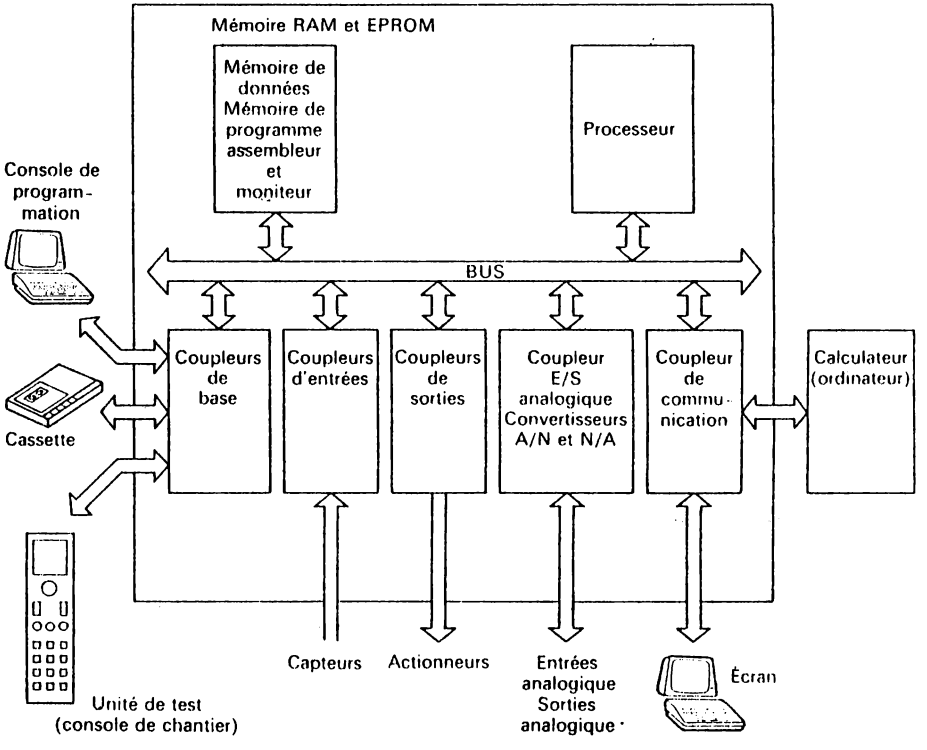


Fig. 13. 2

2. LES CAPTEURS

Tout problème d'automatisme commence par celui de la saisie d'informations provenant du système à automatiser.

Parmi les nombreux systèmes proposés nous choisirons ici les plus simples et, afin de ne pas multiplier les schémas, nous adopterons ceux qui utilisent des contacts secs. On pourra facilement imaginer des montages analogues utilisant un transistor, un contact magnétique, voire même une petite vanne admettant un gaz ou un liquide sous pression.

2.1 Capteurs de pression

On peut être amené à détecter une pression pour obtenir, entre autres :

– l'arrêt automatique d'une machine en cas de baisse de la pression d'huile ou de manque d'air,

– un démarrage autorisé seulement en présence d'une certaine pression d'un fluide déterminé,

– la régulation de la pression d'un fluide entre deux valeurs données,

– etc.

Les dispositifs commercialisés sont appelés « manostats », « pressostats », « contrôleurs de pression », etc...

Ils utilisent en général l'un des principes suivants :

a) Action du fluide sur une membrane (*fig.* 13. 3).

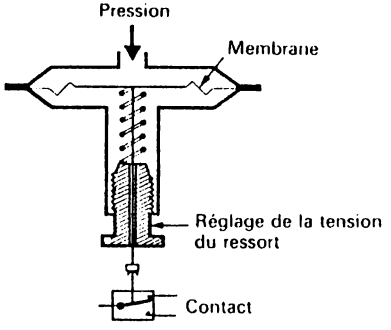


Fig. 13. 3

La grande surface offerte par cette membrane permet de contrôler les faibles pressions (à partir d'un millimètre d'eau).

b) Action du fluide sur un tube de Bourdon (*fig.* 13. 4).

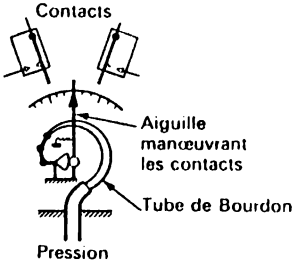


Fig. 13. 4

C'est un dispositif que l'on retrouve dans de nombreux manomètres. Certains d'entre eux comportent des contacts mobiles, disposés sur le cadran lui-même et manœuvrés par le déplacement de l'aiguille.

c) Action sur un tube ondulé (*fig.* 13. 5).

Les applications sont semblables à celles du tube de Bourdon.

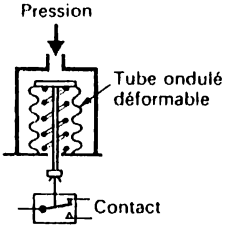


Fig. 13. 5

d) Action sur un piston (*fig.* 13. 6).

C'est un système très simple : il s'agit d'un vérin à simple effet muni d'un ressort taré. Néanmoins les forces de frottement en compromettent la sensibilité.

Avec un piston de faible diamètre il sert à contrôler de fortes pressions.

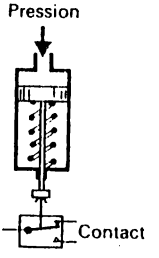


Fig. 13. 6

2.2 Capteurs de température

On imagine facilement l'utilité de tels capteurs, notamment dans le dispositif de sécurité ou la surveillance de la marche d'une chaudière.

Un moyen simple consiste à utiliser la déformation d'un bilame (*fig.* 13. 7) ou d'une spirale.

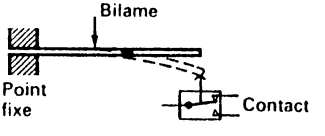


Fig. 13. 7

On peut aussi mettre à profit la dilatation d'un liquide ou l'augmentation de pression d'un gaz, ce qui ramène aux dispositifs précédents.

Notons enfin la possibilité d'utiliser les thermistances et les thermocouples. Ces dispositifs génèrent alors un signal analogique (variation d'un courant, d'une tension) qui est en général lié de façon linéaire à la mesure à enregistrer (ici la température). Pour que cette mesure puisse être utilisée par l'unité centrale de l'automate il faudra transformer la valeur analogique en valeur digitale. Un outil sera indispensable pour cela : le convertisseur analogique-digital.

2.3 Capteurs de vitesse

Il est difficile de contrôler directement une vitesse de translation. On a recours à des capteurs de déplacement d'une part et à une horloge d'autre part. La vitesse sera obtenue en divisant le déplacement par le temps.

En revanche on peut contrôler directement les vitesses de rotation (d'un moteur par exemple) grâce aux systèmes suivants :

a) Utilisation de la force centrifuge (principe du régulateur de Watt). Voir figure 13. 8.

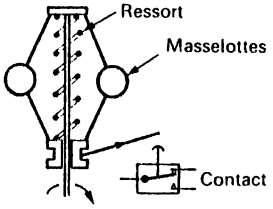


Fig. 13. 8

A une certaine vitesse les masselottes s'écartent, entraînées par la force centrifuge; elles agissent alors sur le contact, provoquant son ouverture ou sa fermeture.

b) Entraînement d'un disque tachymétrique (*fig.* 13. 9).

L'arbre en rotation comporte un aimant situé à très faible distance d'un disque métallique. Ce disque est entraîné par l'aimant et retenu par un ressort en spirale. La manœuvre du contact se produit à partir d'une vitesse donnée de l'arbre.

c) Emploi d'une dynamo tachymétrique.

Une dynamo tachymétrique est un moteur à courant continu que l'on fait fonctionner en génératrice. Plus le rotor tourne vite dans le stator plus la tension aux bornes de la dynamo est élevée, cette tension étant proportionnelle à la vitesse.

Là encore pour exploiter cette information il faudra avoir recours à un convertisseur analogique-digital.

d) Emploi d'un générateur d'impulsions.

Ce capteur sera décrit dans la rubrique des capteurs de position.

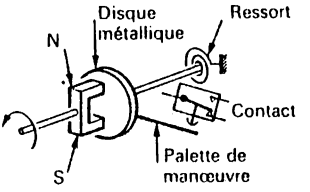


Fig. 13. 9

2.4 Capteurs de force

Ces capteurs utilisent le principe du dynamomètre, c'est-à-dire la déformation d'un ressort.

La figure 13. 10 montre un capteur de force fonctionnant à la traction.

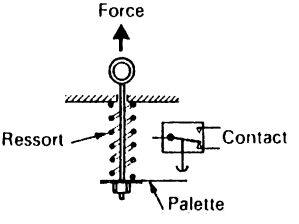


Fig. 13. 10

Sur la figure 13. 11 on mesure l'effet d'une force de compression sur un anneau élastique (qui pourrait être remplacé par un ressort de compression).

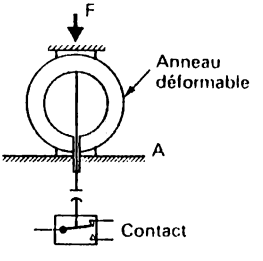


Fig. 13. 11

2.5 Capteurs de position

On peut être amené soit à vérifier la présence d'un objet (un outil par exemple), soit à en déterminer sa position, soit à en mesurer son déplacement.

2.5.1 Capteurs de présence

Qu'il s'agisse de vérifier qu'un outil ou un fil n'est pas cassé, qu'un trou a été usiné, une solution simple consiste à placer directement un palpeur sur l'objet.

Les figures 13. 12 *a*, *b* et *c* donnent quelques exemples de ce type de solution.

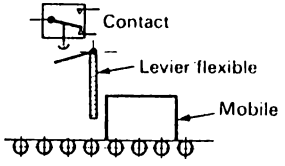


Fig. 13. 12 *a*

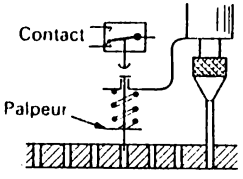


Fig. 13. 12 *b*

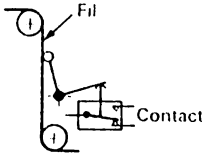


Fig. 13. 12 *c*

D'autres dispositifs peuvent utiliser un rayon lumineux, un dispositif magnétique, etc... (cf. *fig.* 13. 13).

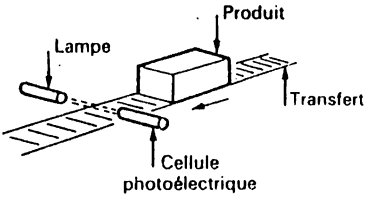


Fig. 13. 13

2.5.2 Capteurs de position proprement dits

On peut mettre à profit la présence de l'objet dans une position déterminée pour provoquer par son passage, la fermeture d'un contact (*fig.* 13. 14).

Dans d'autres applications on pourra, là encore, utiliser un détecteur de proximité, une vanne hydraulique ou pneumatique, une cellule photoélectrique, etc.

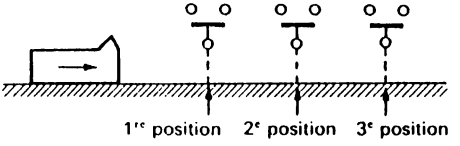


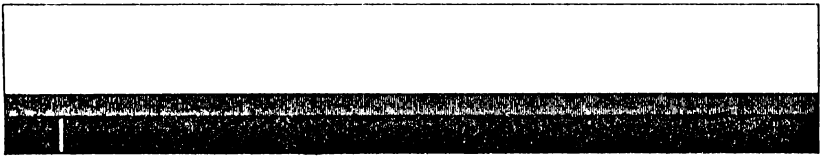
Fig. 13. 14

2.5.3 Capteurs de déplacement linéaire

La mesure de déplacements linéaires peut être faite avec une grande précision par des générateurs d'impulsions.

Il s'agit par exemple d'une règle de verre graduée par un réseau de traits au chrome et fixée sur le mobile. Un système opto-électronique fixe réagit à chaque trait qui coupe le faisceau lumineux et produit des impulsions qu'il suffit de compter à partir d'une origine.

On voit sur la figure 13. 15 la photo d'une telle règle. Le trait blanc à gauche sert à donner le zéro du comptage.



codes

Fig. 13. 15

Une variante de ce principe (*fig.* 13. 16)) permet d'effectuer directement le comptage sur 10 bits, à condition de disposer de dix têtes de lecture. Ces règles codeuses ont un avantage essentiel : il n'est plus nécessaire de « passer par zéro » pour avoir la position précise puisque chaque position est codée directement.

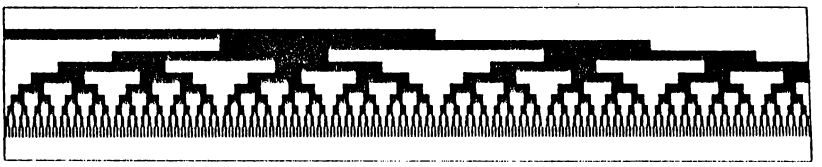


Fig. 13. 16

2.5.4 Capteurs de déplacement relatif

Le même système peut être utilisé pour mesurer des angles de rotation. La règle devient alors un disque gradué ou une « roue codeuse » (*fig.* 13. 17 *a* et *b*).

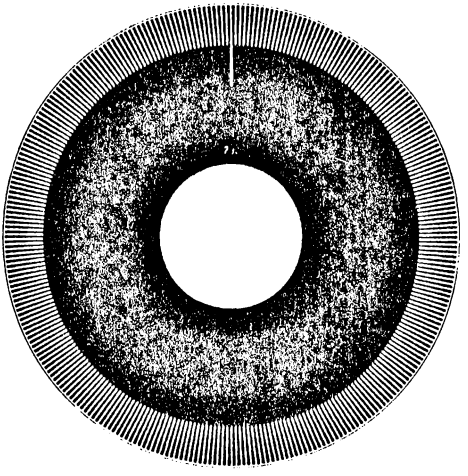


Fig. 13. 17 *a*

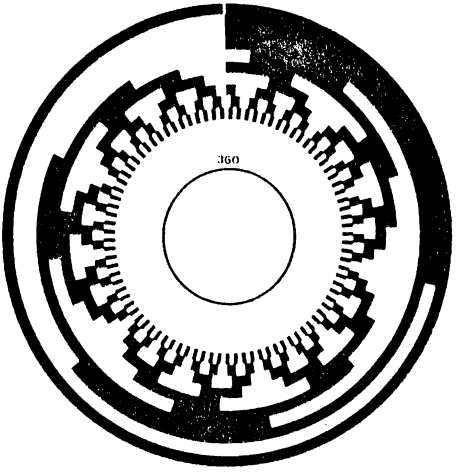


Fig. 13. 17 *b*

3. LES ACTIONNEURS

Recevant des informations du système par l'intermédiaire des capteurs, le processeur va traiter ces informations ce qui est affaire de programme (nous en étudierons la conception générale un peu plus loin), puis il intervient sur ce même système pour déclencher un certain nombre d'actions. Il faut donc qu'il dispose d'« actionneurs ». Il s'agira par exemple d'ouvrir une vanne, de commander le déplacement d'un chariot, de déclencher l'ouverture d'une porte ou d'un volet, etc...

La plupart de ces actionneurs seront dotés d'un moteur réalisant la manœuvre recherchée.

Le courant délivré par le processeur n'étant pas suffisant pour actionner directement ces moteurs il faudra donc disposer de contacts-relais.

Dans un très grand nombre de cas, le processeur enverra donc ses ordres à un relais. Il suffira pour cela d'une seule information binaire.

On pourra alors distinguer différents cas suivant la puissance nécessaire à l'actionneur proprement dit, c'est-à-dire au moteur.

Dans d'autres cas l'actionneur peut requérir différents niveaux de commande.

Nous allons voir quelques exemples de ces différentes commandes.

3.1 Commande d'actionneur directe ou par interrupteur à relais

Si l'actionneur est un voyant lumineux, on peut imaginer une commande directe en utilisant une porte inverseuse à collecteur ouvert (*fig.* 13. 18). Ce schéma est classique et a déjà été utilisé à plusieurs reprises lors des expériences proposées dans cet ouvrage.

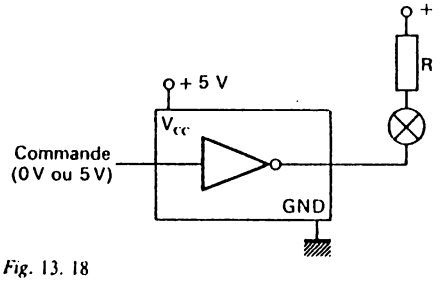


Fig. 13. 18

On peut utiliser également ce schéma (*fig.* 13. 19) pour fermer un relais. Certains relais en effet fonctionnent sous 5 volts et n'absorbent que quelques milliampères. Or une porte TTL de puissance peut, lorsqu'elle est au niveau bas, absorber un courant de 50 mA.

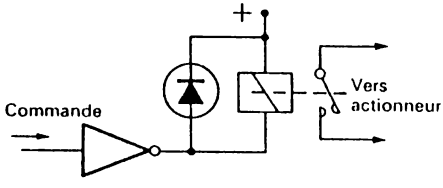


Fig. 13. 19

On notera la diode de protection contre les surtensions inverses provoquées par la coupure du contact.

En dehors de ces quelques cas il faudra opter pour une interface de puissance, c'est-à-dire un étage amplificateur de tension et de courant permettant de délivrer la puissance nécessaire à la bobine d'un relais de puissance.

On a alors recours soit à des transistors soit à des thyristors ou des triacs. Dans ces deux derniers cas on prendra soin d'assurer le déclenchement au passage à la tension nulle pour éviter de perturber la logique de commande par des parasites.

3.2 Commandes quantifiées

Des actionneurs plus «subtils» doivent être commandés soit par un certain nombre d'impulsions soit par des niveaux exprimés par des nombres binaires.

Le moteur pas à pas est le type même d'actionneur à commande par impulsions. Certains de ces moteurs ont un pas angulaire de 1°, c'est-à-dire qu'ils tournent de 1° à chaque impulsion. Leur vitesse sera déterminée par la fréquence de ces impulsions.

On peut aussi avoir à faire à des actionneurs fonctionnant en courant continu et sensibles à une tension ou à un courant délivré par la commande. Dans ce cas, comme la logique du processeur ne peut délivrer que des nombres digitaux, on aura recours à un convertisseur digital-analogique capable de convertir des nombres binaires (en général de 8, 10 ou 12 bits) en un courant proportionnel au nombre binaire affiché en entrée.

Un convertisseur courant-tension est alors souvent nécessaire avant l'interface de puissance. On fait appel pour cela à un amplificateur opérationnel

4. LE GRAFCET

Le programme de l'automate ne peut être écrit avant que l'on ait analysé en détail toutes les fonctions qu'il doit remplir et l'ordre dans lequel il doit les remplir. Cette analyse se fait au moyen d'un outil standardisé qui est le GRAFCET (Graphe de Commande Étapes-Transitions).

En effet, le fonctionnement d'un automate peut être représenté graphiquement par des successions :

– d'*étapes* auxquelles sont associées des actions,

– de *transitions* auxquelles sont associées des «réceptivités» (des informations provenant des capteurs).

4.1 GRAFCET de 1er niveau

Le GRAFCET de premier niveau décrit en clair les différentes actions à effectuer à chaque étape ainsi que les réceptivités conditionnant chacune des transitions d'une étape vers la suivante.

Pour illustrer le principe de ce GRAFCET nous vous proposons l'exemple d'une machine à poinçonner des pièces métalliques.

4.1.1 Schéma du mécanisme à automatiser (fig. 13. 20)

Un plateau tournant permet d'amener sous le poste de poinçonnage les pièces métalliques. Les trois autres positions figurant sur ce plateau correspondent aux postes d'approvisionnement et d'éjection des pièces dont nous ne nous occuperons pas ici.

Les abréviations du schéma correspondent aux actionneurs et capteurs suivants :

- DE : actionneur de descente du poinçon.
- MO : actionneur de montée du poinçon.
- RO : actionneur de rotation du plateau.
- *fh* : capteur de position « poinçon en haut ».
- *fb* : capteur de position « poinçon en bas ».
- *r0* : capteur position initiale du plateau.
- *r1* : capteur position finale du plateau.
- *dcy* : bouton poussoir de départ du cycle.

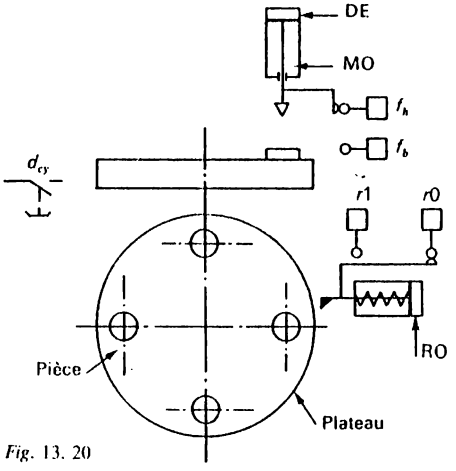


Fig. 13. 20

4.1.2 Fonctionnement

– Le système se trouvant en « position initiale », l'appui sur *dcy* (départ du cycle) ordonne la rotation du plateau.

– Une fois la rotation achevée, le poinçon descend.

– Lorsque le poinçon est arrivé en position basse, il doit remonter.

– Le poinçon étant parvenu en position haute la machine est mise en attente d'un nouveau « départ cycle ».

4.1.3 Établissement du GRAFCET de 1er niveau

Du schéma qui vient d'être proposé au paragraphe précédent se déduit directement le GRAFCET suivant : (voir *fig.* 13. 21).

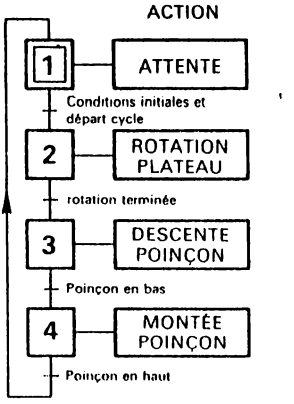


Fig. 13. 21

Étape 1 : étape initiale, attente de l'ordre de départ.

Transition 1-2 : validée par les réceptivités :

a) conditions initiales,

b) ordre de « départ cycle ».

Étape 2 : mise en rotation du plateau.

Transition 2-3 : validée par la fin de rotation du plateau.

Étape 3 : commande de descente du poinçon.

Transition 3-4 : validée par la position basse du poinçon.

Étape 4 : commande la montée du poinçon.

Transition 4-1 : validée par la position haute du poinçon.

Nous constatons que ce GRAFCET est une succession alternée d'étapes et de transitions.

A chaque *étape* se trouvent associées des *actions*.

A chaque *transition* des informations ou *réceptivités* conditionnent le passage à l'étape suivante.

Chaque étape est symbolisée par un carré et repérée par un numéro.

L'étape initiale est représentée par deux carrés concentriques. L'automatisme devra obligatoirement commencer par elle.

4.2 GRAFCET de 2e niveau

Le GRAFCET de deuxième niveau est plus étroitement lié à la nature :

– des actionneurs dont la commande permet d'obtenir l'action à réaliser,

– des capteurs délivrant les informations logiques qui permettent de valider les transitions.

Ce GRAFCET est représenté en figure 13. 22.

On se reportera au schéma de l'automatisme et au paragraphe 4.1.1 pour ce qui concerne la désignation des capteurs et des actionneurs.

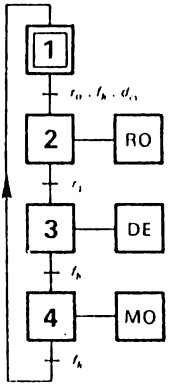


Fig. 13. 22

5.NOTIONS DE SÉQUENCE

Nous allons décrire ici les différentes séquences que l'on peut trouver sur un GRAFCET ainsi que leurs symbolisations.

5.1 Étape

Une étape est une situation dans laquelle le comportement de tout ou partie du système est invariant : les entrées et sorties ne changent pas de valeur.

Rappelons qu'une étape est repérée par un carré et un numéro et qu'une étape activée en début de cycle (ou étape initiale) se représente par un double carré.

Si l'on désire indiquer l'étape active à un instant donné, il suffit de la marquer d'un point à l'intérieur du carré.

5.2 Transition

Une transition indique la possibilité d'évolution d'une étape vers une autre étape. A chaque transition est associée une réceptivité.

Pratiquement une réceptivité sera représentée par une fonction combinatoire de variables d'entrée. Ces variables d'entrée peuvent non seulement traduire des états de fin de course ou de boutons de commande mais aussi le résultat d'une temporisation ou d'une comparaison.

Une réceptivité peut également tenir compte des fronts montants (↑) ou descendants (↓) des variables. Par exemple a ↑ signifiera front montant de la variable *a*.

5.3 Actions associées à une étape

Les actions associées à une étape sont décrites littéralement ou symboliquement dans un ou plusieurs rectangles situés à droite de l'étape (*fig.* 13. 23).

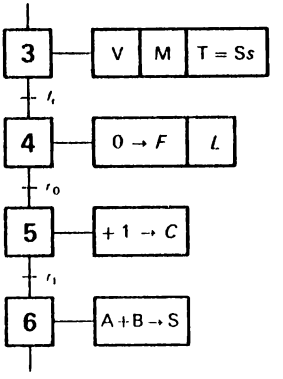


Fig. 13. 23

Ces actions peuvent être par exemple :
- ouvrir une vanne (V)
- faire tourner un moteur (M)
- allumer un voyant (L)
- lancer une temporisation (T)
- mettre 1 dans une mémoire (1 → F)
- mettre 0 dans une mémoire (0 → F)
- incrémenter un compteur (+ 1 → C)
- décrémenter un compteur (− 1 → C)
- effectuer une addition (A + B → S) ou toute autre opération
- etc...

5.4 Actions conditionnelles

L'exécution des actions peut parfois être soumise à d'autres conditions. Ces conditions, notées à côté d'un trait vertical qui est tracé au-dessus du rectangle de l'action, sont représentées par une fonction combinatoire des variables d'entrée.

Le GRAFCET de la figure 13. 24 signifie qu'à l'étape 9 l'avance (AV) ne s'effectue que si la condition c est vraie ($c = 1$); à l'étape 10 la montée (M) est effectuée tant que la temporisation (T) est en cours. Au bout de trois secondes la montée est remplacée par le retour (R).

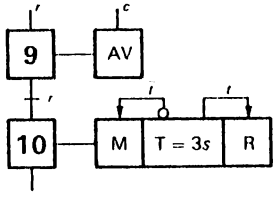


Fig. 13. 24

5.5 Aiguillage

Un aiguillage (ou sélection) entre deux ou plusieurs séquences possibles peut s'effectuer par des conditions de transition exclusives les unes des autres.

Sur la figure 13. 25 l'étape 5 se trouvant active et la réceptivité b étant vraie ($b = 1$), la transition s'effectuera vers l'étape 10.

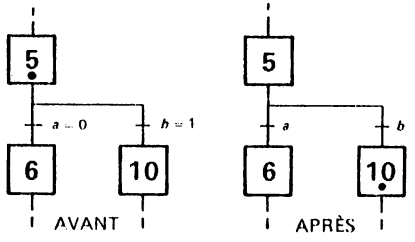


Fig. 13. 25

5.6 Attribution

Elle est illustrée par la figure 13. 26.

Elle s'effectue après que les réceptivités propres à chaque branche aient été vérifiées, ces réceptivités pouvant être, ici, identiques.

Dans l'exemple illustré, lorsque l'étape 14 est active et la réceptivité d vérifiée, la transition s'effectue vers l'étape 9.

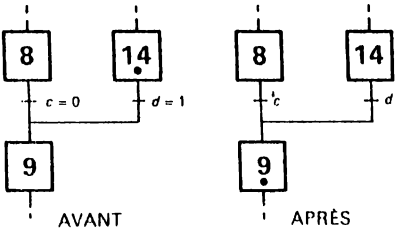


Fig. 13. 26

5.7 Divergence

Deux ou plusieurs séquences peuvent être simultanément activées à partir de la même condition de transition. Deux traits horizontaux parallèles sont alors utilisés pour mettre en évidence ce cas sur un GRAFCET.

La figure 13. 27 indique l'activation simultanée des étapes 8 et 10 à partir de la réceptivité $r = 1$ lorsque l'étape 7 est active.

Le déroulement de chacune des séquences des deux branches a lieu ensuite de façon indépendante.

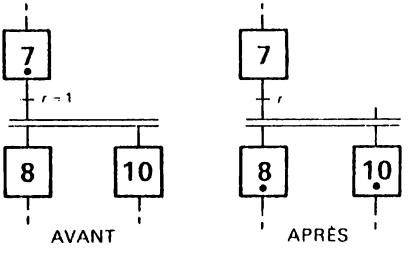


Fig. 13. 27

5.8 Convergence

La convergence (ou jonction) entre plusieurs branches parallèles ne pourra s'effectuer que lorsque toutes les séquences concernées seront terminées et la réceptivité commune vraie (*fig.* 13. 28).

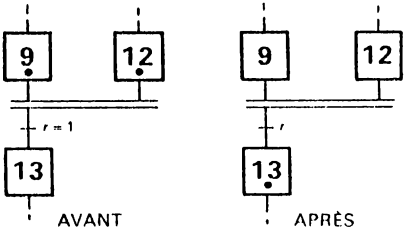


Fig. 13. 28

5.9 Séquences répétées

Si une même séquence doit se répéter plusieurs fois dans un automatisme, on peut la représenter d'une façon simplifiée par un rectangle dont les côtés verticaux sont doublés et où sont indiquées les étapes de début et de fin.

Cette séquence pourra être répétée et accompagnée d'une étape de validation différente à chaque fois.

Ainsi, sur la figure 13. 29, la séquence comportant les étapes 20 à 26 est appelée avec l'étape 14 et avec l'étape 16.

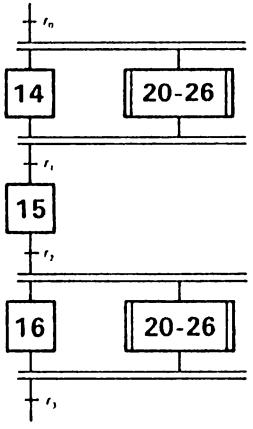


Fig. 13. 29

6. ALÉAS

La méthode GRAFCET peut conduire à certains aléas que nous allons étudier.

6.1 Aléa d'évolution

Un exemple fera comprendre ce type d'aléas : un motoréducteur à deux sens de rotation (*fig.* 13. 30) entraîne un plateau qui doit réaliser un tour complet dans un sens puis dans l'autre.

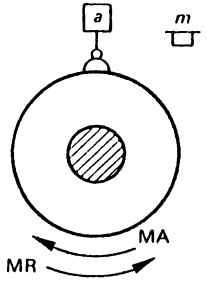


Fig. 13. 30

Le GRAFCET de la figure 13. 31 trace le fonctionnement de principe de ce mécanisme. On remarque qu'une fois la transition effectuée grâce à l'appui sur «*m*», la réceptivité «*a*», qui est en principe attendue après un tour en marche avant du plateau, est déjà présente. La transition vers l'étape 3 se trouve donc validée de façon intempestive et le même phénomène se retrouve pour la transition de l'étape 3 vers l'étape 1.

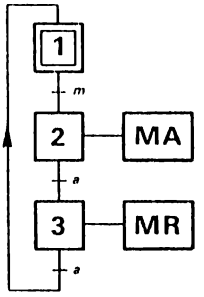


Fig. 13. 31

Deux solutions sont alors possibles pour supprimer cet aléa :

– soit ne prendre en compte l'état de la réceptivité (*a*) qu'après avoir vérifié son état de repos ($\bar{a}$) (cf. *fig.* 13. 32 *a*),

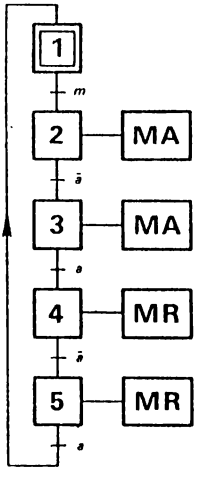


Fig. 13. 32 *a*

— soit prendre en compte le front montant de la réceptivité ($a\uparrow$) (cf. *fig.* 13. 32 *b*).

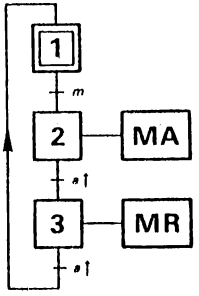


Fig. 13. 32 *b*

6.2 Aléa de bouclage

Cet aléa est illustré très simplement par le cas d'un moteur commandé par marche (*m*) et arrêt (*a*). Dans le cas du GRAFCET de la figure 13. 33 les réceptivités *m* et *a* étant simultanément vraies au moment de la commande *m*, le système fera n'importe quoi.

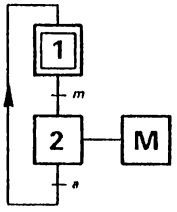


Fig. 13. 33

La solution consiste à rendre les réceptivités exclusives l'une de l'autre, par un GRAFCET à marche prioritaire (*fig.* 13. 34 *a*) ou à arrêt prioritaire (*fig.* 13. 34 *b*).

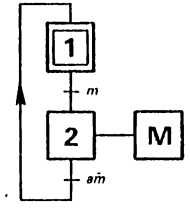


Fig. 13. 34 *a*

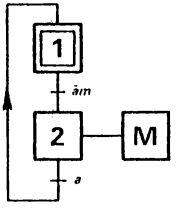


Fig. 13. 34 *b*

6.3 Aléa d'aiguillage

Si les réceptivités conditionnant l'aiguillage ne sont pas exclusives l'une de l'autre on peut provoquer une divergence (*fig.* 13. 35). La solution sera analogue à la solution précédente (*fig.* 13. 36 *a* et *b*).

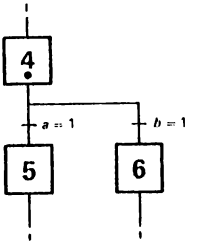


Fig. 13. 35

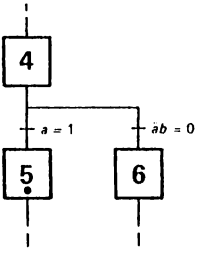


Fig. 13. 36 *a*

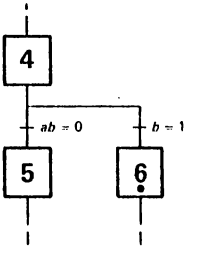


Fig. 13. 36 *b*

6.4 Aléa de non simultanéité

Cet aléa peut se produire lorsqu'une réceptivité fugitive conditionne une convergence.

Par exemple (*fig.* 13. 37) si *a* apparaît fugitivement alors que *b* n'est pas encore présente, la convergence ne se produit pas.

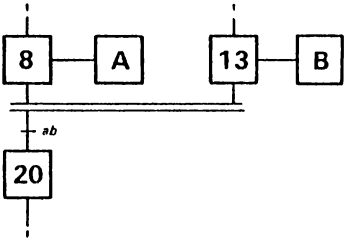


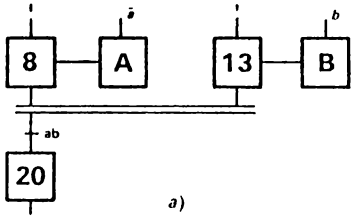
Fig. 13. 37

Deux solutions sont possibles pour remédier à cet aléa :

– Conditionner chacune des actions A et B par leur réceptivité correspondante. Ceci suppose que les actions en question n'ont pas pu être poursuivies au delà des réceptivités et, de plus, que ces actions s'achèvent plus vite que ne retombent les réceptivités (*fig.* 13. 38 *a*).

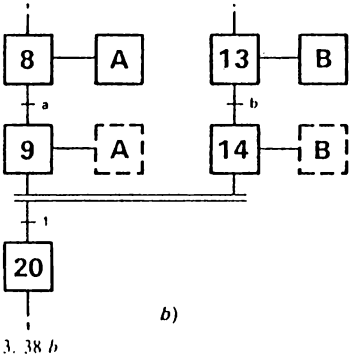
– Ajouter une étape mémorisant l'apparition de chaque réceptivité (*fig.* 13. 38 *b*).

La convergence vers l'étape 20 sera alors systématique une fois les étapes 9 et 14 simultanément actives.



a)

Fig. 13. 38 *a*



b)

Fig. 13. 38 *b*

7. LES TEMPORISATIONS

Une transition peut être le résultat d'une temporisation effectuée à partir d'une certaine étape.

Le GRAFCET devra donc indiquer :

– l'origine de la temporisation (qui sera une étape)

– l'action à effectuer à la fin de la temporisation.

La figure 13. 39 nous montre l'utilisation du temporisateur T en tant que réceptivité :

– L'étape 3 actionne le moteur et le temporisateur T.

A l'issue des dix secondes, « *t* » qui apparaît provoque la transition vers l'étape 4 afin d'ouvrir la vanne, alors que le temporisateur, qui est inutile, est relâché.

– A la suite de l'appui sur « *a* », le temporisateur est à nouveau déclenché à l'étape 5 alors que la vanne est refermée.

– A l'issue des dix secondes, « *t* » provoque la transition vers l'étape 6 (arrêt du moteur, allumage du voyant L et relâchement du temporisateur).

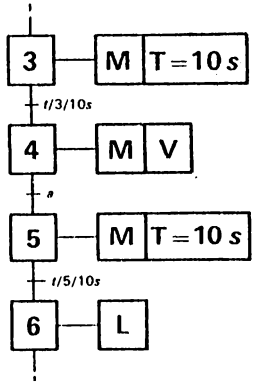


Fig. 13. 39

8. MODES DE FONCTIONNEMENT

Un mode de fonctionnement est une des variantes de déroulement des différents cycles de marche ou d'arrêt de l'automatisme.

8.1 Contrôle des modes de marche

Le contrôle des différents modes de marche peut se définir par un GRAFCET spécial comme celui de la figure 13. 40.

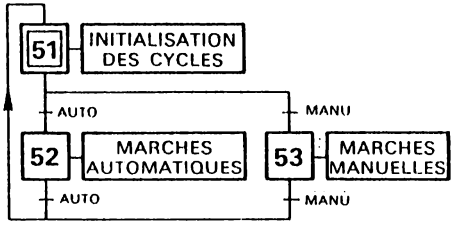


Fig. 13. 40

8.1.1 Initialisation des cycles

L'étape initiale 51 du GRAFCET de contrôle permet de désarmer tous les cycles de l'automatisme.

Ceci permet, notamment au moment de la mise sous tension, d'initialiser les cycles en remettant à 0 l'ensemble des étapes, certaines pouvant se trouver dans un état quelconque.

En outre cette étape permet éventuellement de visualiser l'état initial du système automatisé.

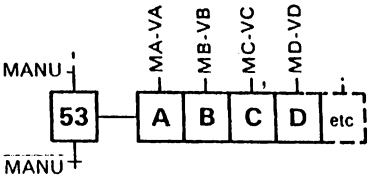


Fig. 13. 41

8.1.2 Marches automatiques

L'étape 52 du GRAFCET de contrôle ordonne le départ des marches automatiques. La transition vers cette étape est conditionnée par la réceptivité «auto» à laquelle on peut ajouter les conditions de position initiale du système.

Le départ des marches automatiques est souvent obtenu par l'armement de la séquence correspondante à l'automatisme en question. Lorsque plusieurs armements sont nécessaires et qu'ils doivent être exécutés dans un ordre déterminé, ils s'effectueront sous le contrôle de plusieurs étapes analogues à l'étape 52.

Les marches automatiques sont considérées comme le fonctionnement normal du système. Plusieurs types de marches automatiques sont envisageables :

- marche en cycles continus,
- marche cycle par cycle,
- marche étape par étape.

Ce dernier type de marche est utile au moment de la mise au point du système.

Dans chaque cas il suffit de prévoir les réceptivités manuelles nécessaires pour relancer les cycles ou les étapes.

8.1.3 Marches manuelles

L'étape 53 du GRAFCET de contrôle (*fig.* 13. 40) permet d'établir les différentes marches manuelles du système. La transition vers cette étape est constituée par la réceptivité «manu».

Dans la plupart des cas, les marches manuelles sont représentées dans l'étape même sous forme d'actions conditionnelles (voir *fig.* 13. 41) :

MA, MB, MC, MD... : boutons poussoirs ordonnant la commande manuelle de la sortie considérée

V_A, V_B, V_C, V_D... : sécurités et verrouillages sur la commande manuelle considérée.

8.2 Contrôle des modes d'arrêt

On voit sur la figure 13. 40 le moyen de revenir, après un fonctionnement automatique ou manuel, à l'étape initiale. Celle-ci provoque l'arrêt systématique des cycles en les désarmant.

Un contrôle de fin de cycle automatique ou de position initiale est possible, avant même de revenir à cette étape.

Dans certaines circonstances, néanmoins, on doit être en mesure de mettre hors service le cycle en cours et d'enclencher un cycle de dégagement ou d'initialisation de la position du système. C'est le cas où apparaîtrait un défaut de fonctionnement du système notamment.

Différents cas d'arrêt sont possibles :

a) Premier cas (*fig.* 13. 42)

Nous avons rajouté au GRAFCET précédent une étape 54 vers laquelle transite le système en cas de défaut ou d'arrêt d'urgence (AT).

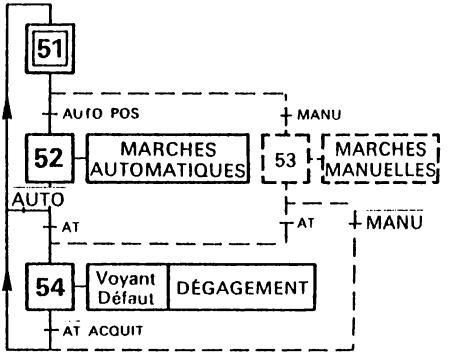


Fig. 13. 42

Cette étape peut, par exemple, commander un avertisseur sonore ou lumineux et déclencher un cycle de dégagement.

Le retour vers l'étape initiale 51 est conditionné par $\overline{AT}$ (défaut disparu) et un éventuel acquittement (ACQUIT).

b) Deuxième cas (fig. 13. 43)

Ici l'étape 54 se contente d'avertir de la présence d'un défaut après avoir provoqué le désarmement des modes de marche.

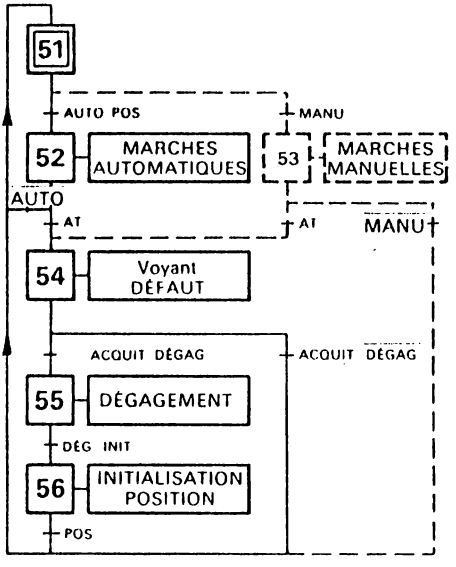


Fig. 13. 43

A partir de là l'opérateur a le choix entre deux solutions :

– soit, après acquittement, revenir directement à l'étape initiale (51) sans effectuer de dégagement (DEGAG), celui-ci pouvant avoir lieu manuellement (étape 53).

– soit, après acquittement, passer par l'étape de dégagement (55) pouvant être suivie d'une étape d'initialisation des positions (56).

9. SYNCHRONISATION DES SÉQUENCES

On peut être amené, pour certains cycles automatiques à synchroniser deux ou plusieurs séquences entre elles.

Supposons une séquence 2 dans laquelle la transition vers une étape 26 ne doit pouvoir s'effectuer qu'une fois l'étape 13 de la séquence 1 activée.

a) Premier principe : utilisation d'une divergence et d'une convergence.

L'illustration en est faite par la figure 13. 44 : la séquence 2 reste bloquée à l'étape 25 tant que l'étape 30 n'est pas active. Or cette étape est activée par la divergence effectuée justement lors de la transition vers l'étape 13 de la séquence 1.

Nous obtenons le résultat souhaité.

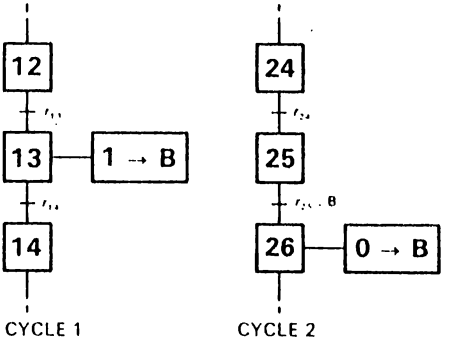


Fig. 13. 44

b) Deuxième principe : utilisation d'un bistable auxiliaire (B) (*fig. 13. 45*).

On reste bloqué à l'étape 25 de la séquence 2 tant qu'un bistable auxiliaire B n'est pas à l'état logique 1. Or ce bistable est enclenché lors de l'arrivée à l'étape 13 de la séquence 1.

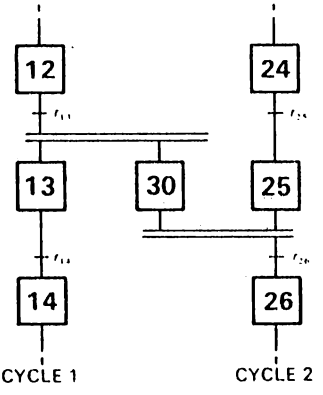


Fig. 13. 45

Il est indispensable de remettre à l'état logique 0 ce bistable une fois la transition qu'il conditionne effectuée. Ceci peut s'effectuer durant l'étape 26. Encore faut-il s'assurer que l'ordre de mise à 1 (étape 13) soit retombé avant celui de remise à 0 (étape 26). Si tel n'était pas le cas, ce bistable devra soit être remis à 0 à une autre étape ne présentant aucune ambiguïté, soit être enclenché sur un front lors de l'étape 23.

10. LA PROGRAMMATION D'UN AUTOMATE

Il existe de nombreux types d'automates programmables et ce nombre ira grandissant d'année en année. Chaque type possède ses propres caractéristiques, tout comme chaque microprocesseur possède les siennes, et en particulier son langage.

Plutôt que d'exposer les règles générales et les variantes de la programmation des automates, étude qui serait fastidieuse et vite dépassée, nous proposons de traiter un exemple simple sur un automate précis pour montrer que cette programmation est très conforme à ce qui a déjà été vu au niveau des microprocesseurs.

10.1 Système à automatiser

Il s'agit de déplacer un mobile du point de départ A vers un point B, de l'y laisser dix secondes puis de l'amener au point D puis de le faire retourner au point A.

Les déplacements sont assurés par deux moteurs, 1 et 2, qui peuvent fonctionner en marche avant ou arrière. Le parcours du mobile est représenté sur la figure 13. 46. Son GRAFCET est représenté par la figure 13. 47.

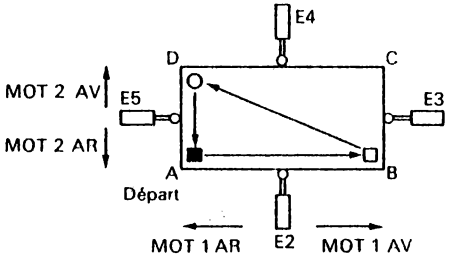


Fig. 13. 46

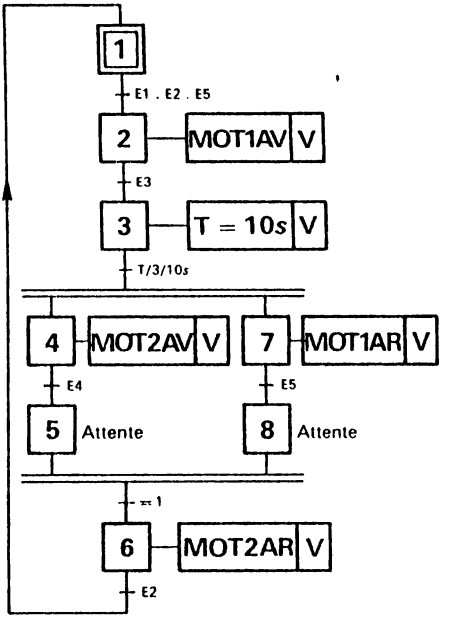


Fig. 13. 47

Liste des sorties :

- Marche avant moteur 1 : MOT 1 AV →
- Marche arrière moteur 1 : MOT 1 AR ←
- Marche avant moteur 2 : MOT 2 AV ↑
- Marche arrière moteur 2 : MOT 2 AR ↓
- Voyant de marche : V

Liste des entrées :

- Bouton de départ : E1
- Fin de course côté AB : E2
- Fin de course côté BC : E3
- Fin de course côté CD : E4
- Fin de course côté DA : E5
- Temporisation : T

10.2 Programmation

L'automate programmable utilisé ici sera un TSX 21 de la Télémécanique, tout autre matériel ayant pu être pris en exemple sur ce cas.

On aura, avant tout, à affecter des adresses de mémoire aux données d'entrée et de sortie :

a) Affectation des adresses d'entrée

Entrée	Adresse
E1	200
E2	201
E3	202
E4	203
E5	204
T	771

b) Affectation des adresses de sortie

Sortie	Adresse
MOT 1 AV	230
MOT 1 AR	231
MOT 2 AV	232
MOT 2 AR	233
V	234

On peut alors traduire le GRAFCET de la figure 13. 47 en un GRAFCET appliqué au TSX 21 et notamment aux adresses qui viennent d'être affectées. On utilise le pas à pas 000 à 004 et les variables de travail 100 et 101 (*fig.* 13. 48).

Il ne reste plus qu'à établir le programme :

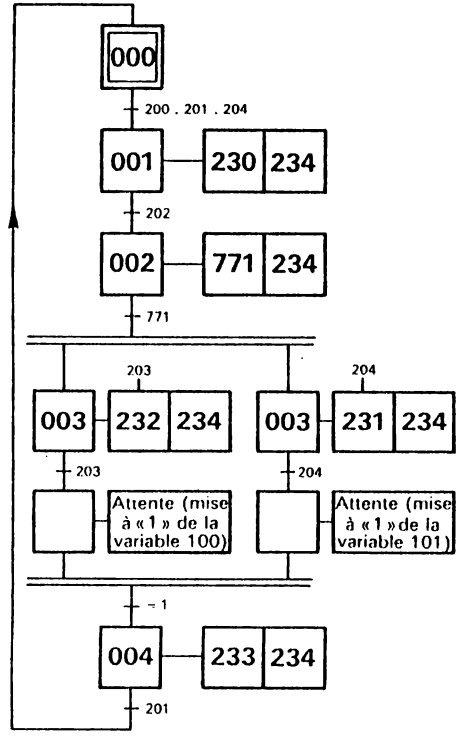


Fig. 13. 48

N° de ligne	Code opération	Adresse de la variable	Commentaires
0001	L	000	charger l'étage 000
0002	A	200	ET entrée 200
0003	A	201	ET entrée 201
0004	A	204	ET entrée 204
0005	=	001	Activer étape 001 ⇒ désactiver étape 000
0006	L	001	charger l'étape 001
0007	A	202	ET entrée 202
0010	=	002	Activer étape 002 ⇒ désactiver étape 001
0011	L	002	charger l'étape 002
0012	A	771	ET temporisation 771 (10 s)
0013	=	003	Activer étape 003 ⇒ désactiver étape 002
0014	L	003	charger l'étape 003
0015	A	203	ET entrée 203
0016	=	100	mettre à « 1 » la variable 100
0017	L	003	charger l'étape 003
0020	A	204	ET entrée 204
0021	=	101	mettre à « 1 » la variable 101
0022	L	100	charger la variable 100
0023	A	101	ET la variable 101
0024	=	004	Activer étape 004 ⇒ désactiver étape 003

N° de ligne	Code opération	Adresse de la variable	Commentaires
0025	L	004	charger l'étape 004
0026	A	201	ET l'entrée 201
0027	=	000	Activer l'étape 000 ⇒ désactiver l'étape 004
0030	L	001	
0031	=	230	Affecter 001 à la sortie 230
0032	L	002	Affecter 002 à la tempo 771
0033	=	771	
0034	L	003	
0035	AN	203	Affecter 003 ET l'INVERSE de 203 à la sortie 232
0036	=	232	
0037	L	003	
0040	AN	204	Affecter 003 ET l'INVERSE de 204 à la sortie 231
0041	=	231	
0042	L	004	Affecter 004 à la sortie 233
0043	=	233	
0044	L	001	Affecter 001 ou
0045	O	002	002 ou
0046	O	003	003 ou
0047	O	004	004 à la sortie 234
0050	=	234	

EXERCICES

Nous vous demandons d'établir le GRAFCET d'un automatisme concernant un malaxeur dont le fonctionnement doit être le suivant (voir figure E. 13. 1).

L'action sur le bouton poussoir MCY déclenche le cycle qui commence par le remplissage simultané des deux trémies avec mise en route du chauffage de la trémie 2. En fait, ce chauffage commence 3 secondes après le début du remplissage de la trémie 2 et il est contrôlé par un thermostat.

Dès que la trémie 1 est pleine, elle est vidée dans le malaxeur qui se met en route.

Dix secondes après la vidange complète de la trémie 1 on vidange la trémie 2 si elle est pleine et à bonne température. On malaxe ensuite l'ensemble pendant 15 secondes (après vidange complète de la trémie 2).

Enfin on vide le malaxeur et le cycle peut recommencer.

Les vidanges et remplissages se referment dès que l'ordre d'ouverture disparaît.

Liste des entrées (boutons-poussoirs et capteurs).

MCY : poussoir marche cycle
THE : thermostat (contact fermé si la température est atteinte)
T1P : trémie N° 1 pleine
T1V : trémie N° 1 vide
T2P : trémie N° 2 pleine
T2V : trémie N° 2 vide
VF1 : vidange trémie N° 1 fermée
VF2 : vidange trémie N° 2 fermée
VFA : vidange malaxeur fermée
MAV : malaxeur vide
temporisation 3 secondes
temporisation 10 secondes
temporisation 15 secondes

Remarque : les entrées sont toutes de nature « tout ou rien ».

Liste des sorties (actionneurs).

RT1 : remplissage trémie N° 1
RT2 : remplissage trémie N° 2
MAL : malaxage
CHA : chauffage
VT1 : vidange trémie N° 1
VT2 : vidange trémie N° 2
VMA : vidage malaxeur.

Remarque : Pour simplifier on ne considèrera ni les boutons-poussoirs « arrêt d'urgence », etc. ni les différents voyants pouvant visualiser la marche de l'automatisme.

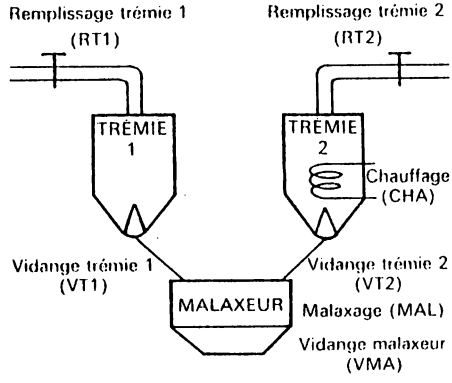


Fig. E. 13. 1

RÉPONSES AUX EXERCICES

Chapitre 2

1. En regroupant le premier et le troisième terme on obtient

$$A\bar{B}CD + \bar{A}\bar{B}CD = \bar{B}CD$$

De même en regroupant le deuxième et le quatrième :

$$\bar{E}B\bar{C}D + EB\bar{C}D = B\bar{C}D$$

et

$$\bar{B}CD + B\bar{C}D = BD$$

l'ensemble des termes est donc équivalent à BD.

2. *a*) Le tableau de Karnaugh correspondant à la condition d'imparité se présente sous forme d'un damier (figure S. 2. 1).

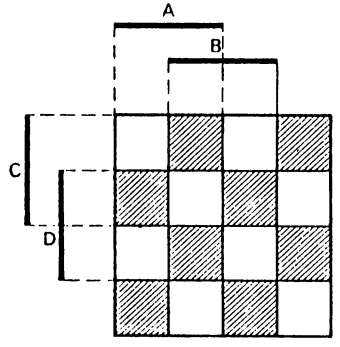


Fig. S. 2. 1

b) En explicitant chaque case hachurée de ce tableau de Karnaugh de gauche à droite en commençant par le haut, on obtient

$$S = A\bar{B}C\bar{D} + \bar{A}\bar{B}CD + A\bar{B}CD + \bar{A}BCD$$
$$+ AB\bar{C}D + \bar{A}B\bar{C}D + AB\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D}$$

On sait aussi (*cf.* § 2.5) que cette relation peut s'écrire

$$S = A \oplus B \oplus C \oplus D$$

c) En utilisant chacune de ces relations on peut établir les schémas de circuit qui y correspondent (figures S. 2. 2 et S. 2. 3).

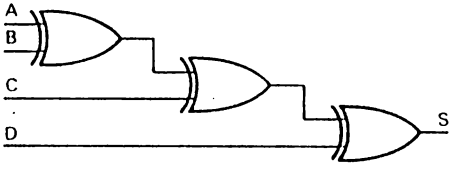


Fig. S. 2. 2

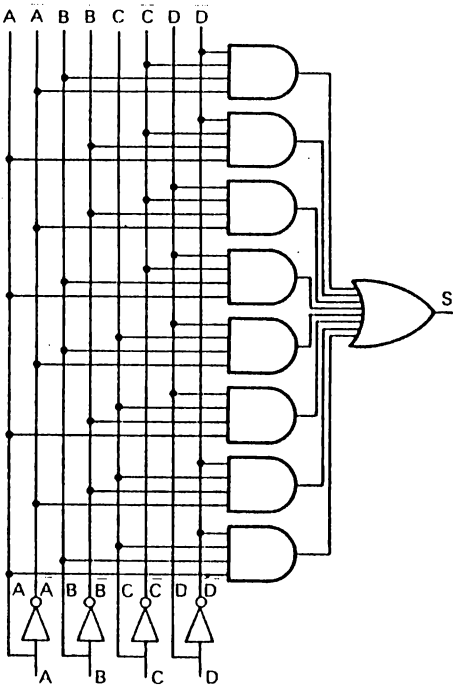


Fig. S. 2. 3

3. On obtient le tableau de Karnaugh suivant (figure S. 2. 4 *a*).

La relation S peut donc s'écrire sous la forme

$$S = AC + AD + AB + BCD$$

ou

$$S = A(B + C + D) + BCD$$

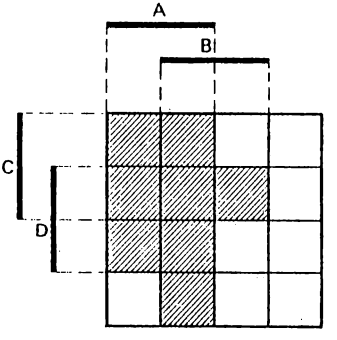


Fig. S. 2. 4 *a*

D'où la réalisation de la «machine à voter» (figure S. 2. 4 *b*).

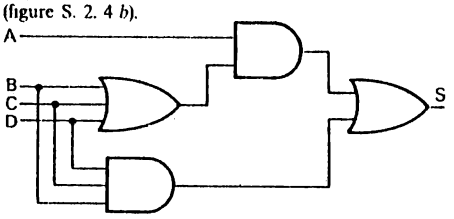


Fig. S. 2. 4 *b*

Chapitre 3

2. Il suffit de construire un chenillard à cinq LED et de prélever, en sortie Q de l'une des cinq bascules, le signal qui sera connecté sur l'entrée H d'une sixième bascule 7474 montée en diviseur par deux.

En utilisant chaque sortie Q_1 à Q_5 pour activer l'entrée horloge de cinq bascules D, on obtiendra cinq signaux identiques, mais décalés chacun du dixième de leur période. De quoi réaliser un nouveau chenillard où la lumière de cinq ampoules se déplace régulièrement.

Chapitre 4

1. *a*) Les quinze valeurs de DCB et A étant dans l'ordre 0000 puis 0001, puis 0010, puis 0011 etc. (numérotation binaire pure), on constituera le premier outil grâce à la table de Karnaugh de la figure S. 4. 1 comportant les valeurs, traduites en hexadécimal, de chacune des 16 cases (ceci a déjà été exposé au § 2.3.1).

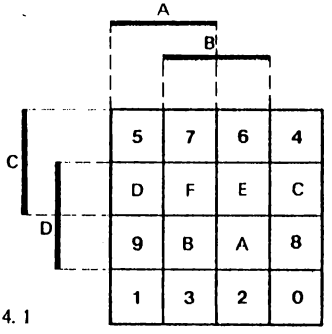


Fig. S. 4. 1

b) Pour plus de facilité dans le raisonnement, il est également bon de construire la même table de Karnaugh avec, pour chaque case, l'expression correspondante en A, B, C, D ou $\overline{A}$, $\overline{B}$, $\overline{C}$ et $\overline{D}$ (*fig.* S. 4. 2).

De plus chacune de ces cases recevra une appellation facile à retrouver : X_{yz} désignera la case de rang *y* et de colonne *z*.

Ainsi

X_{32} représente la case du troisième rang et de la deuxième colonne.

$X_{3\,7}$ représente $AB\overline{C}D$ et correspond au chiffre hexadécimal « B ».

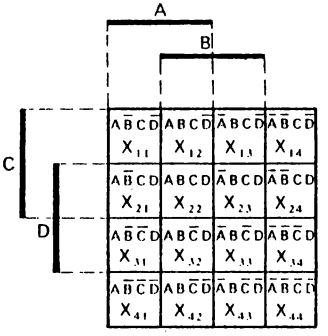


Fig. S. 4. 2

c) Le troisième outil est alors la table des valeurs de *a*, *b*, *c* etc. pour chacun des affichages. Ces valeurs sont déduites directement du symbole affiché lorsque l'on connaît la désignation de chaque segment (*cf.* § 2.3.1).

La figure S. 4. 3 reproduit cette table.

d) A l'aide des outils S. 4. 1 et S. 4. 3, il est aisé alors de remplir des tables de Karnaugh pour chacun des 7 segments *a* à *g*, en marquant d'une croix les seules cases où le segment doit être éteint ($a = 0$ ou $b = 0$ etc.).

	0	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F
a	1	0	1	1	0	1	1	1	1	1	1	0	1	0	1	1
b	1	1	1	1	1	0	0	1	1	1	1	0	0	1	0	0
c	1	1	0	1	1	1	1	1	1	1	1	1	0	1	0	0
d	1	0	1	1	0	1	1	0	1	1	0	1	1	1	1	0
e	1	0	1	0	0	0	1	0	1	0	1	1	1	1	1	1
f	1	0	0	0	1	1	1	0	1	1	1	1	1	0	1	1
g	0	0	1	1	1	1	1	0	1	1	1	1	0	1	1	1

Fig. S. 4. 3

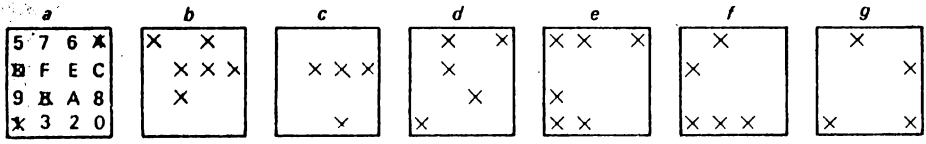


Fig. S. 4. 4

Par exemple, le segment *a* doit être éteint pour l'affichage du 1, du 4, du B et du D. On porte donc une croix dans les cases marquées 1, 4, B et D sur la table de Karnaugh de la figure S. 4. 1 et on obtient la table n° 1 de la figure S. 4. 4.

e) A partir de ce nouvel outil (*fig.* S. 4. 4) et de la figure S. 4. 2, il est facile de donner, pour chaque segment, l'expression qui relie son allumage aux valeurs de ABC et D.

On retiendra que l'afficheur est à anode commune, et que les segments seront éclairés lorsque leur cathode de commande sera au niveau bas. C'est donc les valeurs de $\bar{a}$, $\bar{b}$, $\bar{c}$ etc. qu'il faut trouver. Ces valeurs correspondent exactement aux cases marquées d'une croix à la figure S. 4. 4. D'où

$$\bar{a} = X_{14} + X_{21} + X_{32} + X_{41}$$
$$\bar{b} = X_{11} + X_{13} + X_{22} + X_{23} + X_{24} + X_{32}$$
$$\bar{c} = X_{22} + X_{23} + X_{24} + X_{43}$$
$$\bar{d} = X_{12} + X_{14} + X_{22} + X_{33} + X_{41}$$
$$\bar{e} = X_{11} + X_{12} + X_{14} + X_{31} + X_{41} + X_{42}$$
$$\bar{f} = X_{12} + X_{21} + X_{41} + X_{42} + X_{43}$$
$$\bar{g} = X_{12} + X_{24} + X_{41} + X_{44}$$

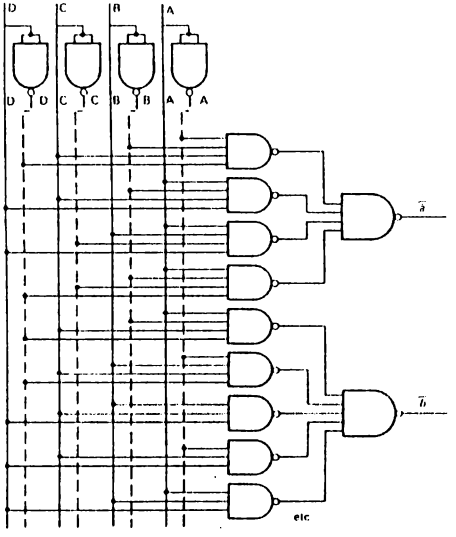


Fig. S. 4. 5

En connaissant la valeur de chaque X_{yz} grâce au tableau de la figure S. 4. 2 on peut en déduire les valeurs des segments (ou plutôt leur complément à 1) :

$$\bar{a} = \bar{A}B\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D}$$
$$\bar{b} = A\bar{B}C\bar{D} + A\bar{B}C\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D + \bar{A}\bar{B}CD + A\bar{B}\bar{C}\bar{D}$$

etc.

f) A partir de là on peut être tenté de simplifier les expressions.

Par exemple, pour $\bar{b}$ les expressions $A\bar{B}C\bar{D} + A\bar{B}CD$ se simplifient en $B\bar{C}D$ et $A\bar{B}\bar{C}D + A\bar{B}C\bar{D}$ donnent $A\bar{C}D$, et encore $A\bar{B}CD + A\bar{B}\bar{C}D$ donnent $A\bar{B}D$, ce qui donne

$$\bar{b} = A\bar{B}C\bar{D} + \bar{A}B\bar{C}D + B\bar{C}D + A\bar{C}D + A\bar{B}D$$

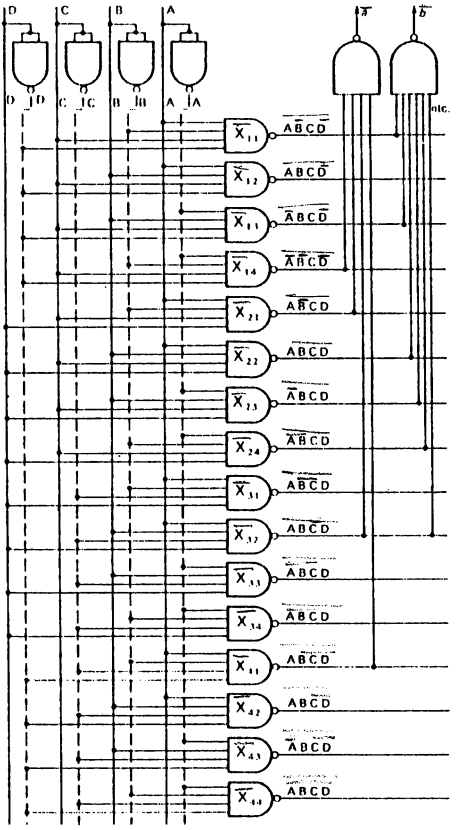


Fig. S. 4. 6

g) Une telle voie est possible, et elle conduira à construire un schéma du type de la figure S. 4. 5 où chaque commande de segment est bâtie, grâce à des portes, directement à partir des valeurs de A, B, C, D, $\overline{A}$, $\overline{B}$, $\overline{C}$ et $\overline{D}$.

h) Une autre solution, plus systématique, et qui, en fin de compte va économiser le nombre des portes à utiliser, consiste à constituer d'abord toutes les valeurs de X_{12} à X_{44}, ou plus exactement de $\overline{X}_{12}$ à $\overline{X}_{44}$, les portes étant des NAND.

C'est à partir de ces valeurs que, avec une seule porte NAND supplémentaire, on obtiendra soit $\bar{a}$, soit $\bar{b}$, soit $\bar{c}$ etc.

En effet, on peut exprimer $\bar{a}$ de deux façons :

$$\bar{a} = X_{14} + X_{21} + X_{32} + X_{41}$$

ou

$$\bar{a} = \overline{X_{14} \cdot X_{21} \cdot X_{32} \cdot X_{41}}$$

La figure S. 4. 6 montre le principe de cette réalisation en se limitant aux sorties $\bar{a}$ et $\bar{b}$.

Il faudra, en tout, vingt-sept portes NAND pour réaliser l'ensemble du décodeur.

Chapitre 5

2. En partant de la configuration 21 (10000) on constate que l'on peut obtenir la configuration 20 par décalage à gauche et rajout d'un 0, puis la configuration 19 par un nouveau décalage à gauche et rajout d'un 1, etc.

Toutes ces configurations peuvent donc être obtenues par la séquence suivante :

(0)0000111101110011010 10000 ←

Le premier 0 à gauche est sans intérêt (division par 1).

Il suffira de conformer le contacteur à glissière à ces valeurs. Toues les entrées devront être raccordées au + 5 V par une résistance de 1 kΩ, leur contact avec le contacteur les mettant au niveau 0. D'où la forme du contacteur (*fig.* S. 5. 1).

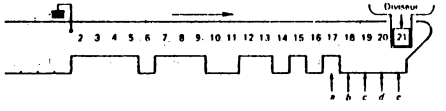


Fig. S. 5. 1

3. *a*) Le signal H_1 se renouvelle toutes les huit périodes de H_0. On peut donc utiliser un multiplexeur à huit entrées et une sortie, commandé par un compteur à trois sorties commandé par l'horloge.

On pourra utiliser le 74151 ou le 74251 comme multiplexeur. Les huit données d'entrée fixes seront donc 10101110 ou toute autre combinaison obtenue par permutation circulaire de ce groupe de valeurs.

L'utilisation d'un registre à décalage nécessite l'utilisation de deux 74195 et deux seulement. En effet, dans le cas présent, on bénéficie du fait que l'on connaît les données d'entrée *abcdefgh*. Il n'est donc pas nécessaire de créer un zéro marqueur. Il suffit d'utiliser un zéro de la séquence de l'horloge H_1 que l'on veut obtenir. On peut donc précâbler les entrées *abcde* sous les formes suivantes (qui sont équivalentes) :

(1) 01110101
(2) 01010111
(3) 01011101

Une analyse de ces trois possibilités doit vous amener à préférer la solution (2). C'est en effet celle où l'on peut limiter à trois les sorties Q qui vont permettre de redéclencher le chargement.

En effet, les différents états des sorties Q lors du décalage seront les suivantes.

	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G	Q_H
Chargement →	0	1	0	1	0	1	1	1
	1	0	1	0	1	0	1	1
	1	1	0	1	0	1	0	1
	1	1	1	0	1	0	1	0
	1	1	1	1	0	1	0	1
	1	1	1	1	1	0	1	0
	1	1	1	1	1	1	0	1
Mise à 0 de Shift/Load	1	1	1	1	1	1	1	0

L'état de remise à zéro de «Shift/Load» sera déterminé par les trois sorties Q_E, Q_F et Q_G, au seul moment où ces trois valeurs sont ensemble à 1 (porte NAND à trois entrées).

On obtient donc le résultat par le montage de la figure S. 5. 2.

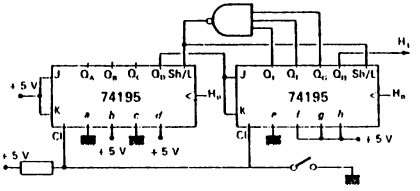


Fig. S. 5. 2

On constatera en outre que le signal H_1 peut être prélevé soit sur Q_H, soit sur Q_G, Q_F ou Q_E, chaque signal étant décalé par rapport au précédent d'une période de l'horloge H_0.

b) On peut obtenir le même résultat en n'utilisant que cinq bascules des circuits 195, la sortie Q_E étant sur $\overline{K}$ du premier circuit et $\overline{Q}_E$ sur J.

On utilisera la séquence allant des configurations 21 à 14, c'est-à-dire

	Q_A	Q_B	Q_C	Q_D	Q_E
Chargement	1	0	0	0	0
	0	1	0	0	0
	1	0	1	0	0
	0	1	0	1	0
	1	0	1	0	1
	1	1	0	1	0
	0	1	1	0	1
Remise à 0 de Sh/L	0	0	1	1	0

On constate que la remise à zéro de «Shift/Load» peut se faire par une porte NAND à deux entrées seulement, correspondant à Q_C et Q_D.

On obtiendra le signal H_1 par la sortie Q_D inversée ($\overline{Q_D}$).

Le montage correspond à la figure S. 5. 3.

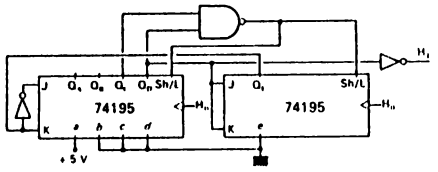


Fig. S. 5. 3

On pourra aussi inverser toute la logique de ce montage :

- en branchant Q_F sur J et $\overline{Q_F}$ sur K,
- en chargeant les valeurs 0 1 1 1 1,
- en disposant un OR à deux entrées sur Q_C et Q_D.

On recueillera alors le signal H_1 sur Q_D sans qu'il soit nécessaire de l'inverser.

Il suffit de décaler le signal 0 dans un registre à 7 bascules chargées de la valeur initiale 0 1 1 1 1 1 1.

Après six décalages la valeur 0 apparaissant dans le registre de droite doit provoquer le chargement, suivant le cycle suivant :

	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F	Q_G	
Chargement	0	1	1	1	1	1	1	
1er décalage	1	0	1	1	1	1	1	
2e	1	1	0	1	1	1	1	
3e	1	1	1	0	1	1	1	
4e	1	1	1	1	0	1	1	
5e	1	1	1	1	1	0	1	
6e	1	1	1	1	1	1	0	→ déclenchement
7e chargement	0	1	1	1	1	1	1	du chargement

Il suffira de brancher les sorties sur $Q_A Q_B \ldots Q_G$ pour obtenir le décodeur.

La sortie Q_G doit simplement être raccordée aux commandes «Shift/Load» des deux 195.

La solution est représentée sur la figure S. 5. 4 (on notera la nécessité de démarrer par un «Clear» pour éviter la configuration 1 1 1 1 1 1 1).

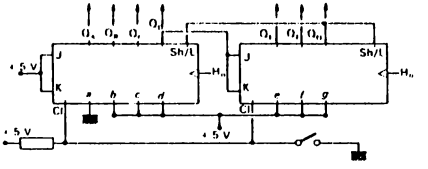


Fig. S. 5. 4

Chapitre 6

1. On disposera huit circuits RAM, branchés en parallèle sur le même bus d'adresses à 10 lignes.

On pourra ainsi, par une même adresse, identique pour chaque circuit, lire ou écrire un mot de huit bits, chaque RAM fournissant ou enregistrant une valeur binaire.

Il faudra que les ordres $\overline{CE}$ et R/$\overline{W}$ parviennent simultanément aux huit circuits. Ces ordres, là encore, seront acheminés en parallèle à partir de fils communs.

Enfin, pour éviter des court-circuits sur le bus de données (qui doit être bi-directionnel) il faudra disposer d'un système de doubles portes «trois états».

L'ordre R doit autoriser la circulation des informations depuis les RAM vers l'unité de calcul et l'interdire dans l'autre sens, et vice-versa pour l'ordre W d'écriture.

On pourra utiliser pour cela deux circuits du type 74243 (voir *fig.* S. 6. 1). En réunissant les broches 1 et 13 à une seule commande on constate qu'un niveau 0 sur cette commande autorise la circulation de données dans le sens A vers B et l'interdit dans le sens B vers A. L'ordre 1 autorise de B vers A et interdit de A vers B.

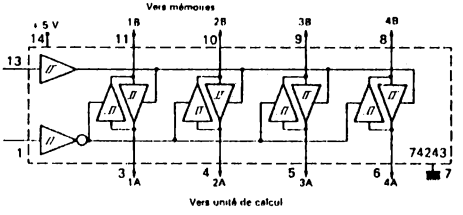


Fig. S. 6. 1

Si l'ordre R/$\overline{W}$ est appliqué à cette commande unique, il faut donc mettre en relation les entrées A avec l'unité de calcul et les entrées B avec les circuits de mémoire.

Le schéma général d'implantation est alors celui de la figure S. 6. 2.

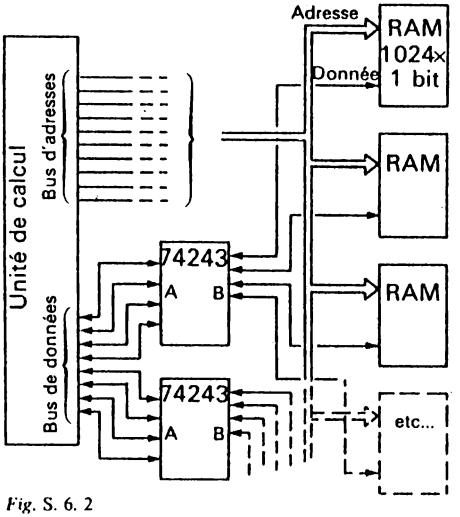


Fig. S. 6. 2

Chapitre 7

1. 111111 + 101111 = 1101110

2. $27_{10} = 0010\ \ 0111$

$+$

$39_{10} = 0011\ \ 1001$

0101 1 0000

+ 1 0110 (+ 6 en correction)

Résultat : 0110 (1)0110 = 66_{10}

3. *a*) Impossible, ce nombre n'existe pas en BCD.

b) 1001 1001 = 99_{10}.

99 −	1	=	98
98/2 = 49 −	1	=	48
48/2 = 24 −	0	=	24
24/2 = 12 −	0	=	12
12/2 = 6 −	0	=	6
6/2 = 3 −	1	=	2
2/2 = 1 −	1	=	0

Résultat = 1100011

Autre méthode : le nombre de gauche comptabilise le nombre de «dizaines» décimales (1010 en binaire)

$N_2 = 1001 \times 1010 + 1001$

la multiplication 1010 × 1001

1010

1010

donne pour résultat 1011010

à additionner à + 1001

Résultat : N = 1100011

4. *a*) 11111 = 100000 − 1

= 32 − 1 = 31 en décimal

Soit 0011 − 0001 en BCD

b) 1011001 = 1000000 = 64

+ 10000 = 16

+ 1000 = 8

+ 1 = 1

89

Soit 1000 − 1001 en BCD

5.

111111

× 111111

111111

111111

111111

111111

111111

111111

111110000001

Autre solution : 111111 = 1000000 − 1

La formule $(a - b)(a + b) = a^2 - b^2$ s'applique quel que soit le système de numération. On utilisera ici la formule $(a - 1)(a + 1) = a^2 - 1$:

$$a^2 = (a - 1)(a + 1) + 1$$

$a^2 = 111110 \times 1000000 + 1$

$a^2 = 111110000001$

6. Pour compter les adresses binaires de 0 à 255 il faut pouvoir disposer de huit bits. Deux compteurs binaires 74161 mis en cascade suffiront.

Pour compter de 0 à 255 en BCD il faut trois compteurs BCD à 4 bits (unités décimales, dizaines et centaines).

La structure de liaison des compteurs avec les trois mémoires MTS 4043 va de soi (*fig.* S. 7. 1).

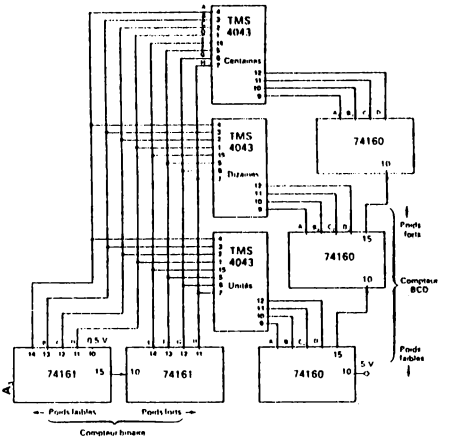


Fig. S. 7. 1

(On portera son attention sur le numéro des broches à relier.)

Les problèmes les plus délicats à résoudre concernent la synchronisation de l'ensemble, le déclenchement et l'arrêt des différents signaux de commande.

Synchronisation des compteurs : Pour que les compteurs démarrent bien en même temps, il faut relier les commandes «$\overline{\text{Clear}}$» à un même circuit suivant un principe déjà vu : le commutateur I fermé maintient les compteurs à 0, ouvert il permet un démarrage synchrone.

Création du signal d'écriture : Observons le chronogramme de l'horloge et des sorties des compteurs binaires adressant les mémoires (*fig.* S. 7. 2). La commande $\overline{W}$ des mémoires doit intervenir après l'application des adresses et il est prudent qu'elle ne dure pas jusqu'à l'apparition d'une nouvelle adresse.

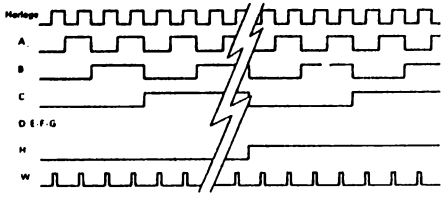


Fig. S. 7. 2

On peut déclencher ce signal sur chaque front montant de l'horloge, à condition d'« amaigrir » le créneau d'horloge en utilisant le dispositif de la figure S. 7. 3 (filtre passe-haut). La petite impulsion qu'il produit, même si elle n'est pas visible sur l'écran de l'oscilloscope, est largement suffisante pour assurer l'écriture en mémoire.

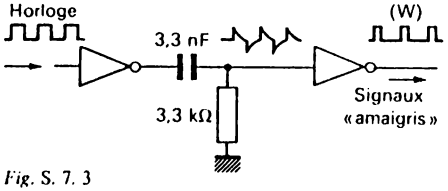


Fig. S. 7. 3

L'inverseur de sortie est à collecteur ouvert. C'est pourquoi il est équipé d'une résistance de charge.

Arrêt du processus d'écriture (voir schéma général S. 7. 5) : Le cycle d'écriture doit durer pendant tout le temps de comptage des 74161 de 0 à 256; il doit s'arrêter ensuite.

Étudions à nouveau le chronogramme de la figure S. 7. 2 et examinons la sortie H (bit de poids le plus fort des sorties binaires des 74161). Ce signal H ne comporte qu'une période. Si, après l'avoir inversé, nous en faisons le signal d'horloge d'une bascule D, cette bascule changera d'état à la fin du cycle de comptage (front positif de $\overline{H}$).

Ce changement d'état pourra être utilisé pour inhiber l'horloge parvenant aux compteurs grâce à la sortie $\overline{Q}$ de cette bascule D (un circuit 7474).

Pour la synchronisation, il faut que cette bascule D soit commandée par le même « Clear » que les compteurs.

Ce montage présente toutefois un risque : au moment où l'on ouvre le commutateur I pour faire démarrer le cycle on peut avoir basculement de la bascule D, une impulsion parasite lui parvenant simultanément.

Pour être sûr que cette bascule ne changera pas d'état, il faut que la fin du signal Clear soit retardée et survienne après le passage de cette éventuelle impulsion.

Nous introduirons donc, dans le circuit « Clear » de la bascule, un système de retardement grâce à un filtre passe-bas. Celui-ci introduit un retard d'environ 1 microseconde sur l'impulsion CE, ce qui suffit à maintenir la bascule en son état ($Q = 0$, $\overline{Q} = 1$) jusqu'à la fin du cycle (*fig.* S. 7. 4).

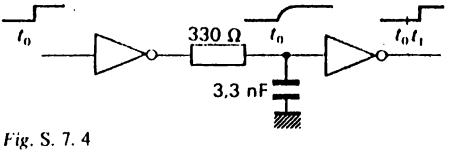


Fig. S. 7. 4

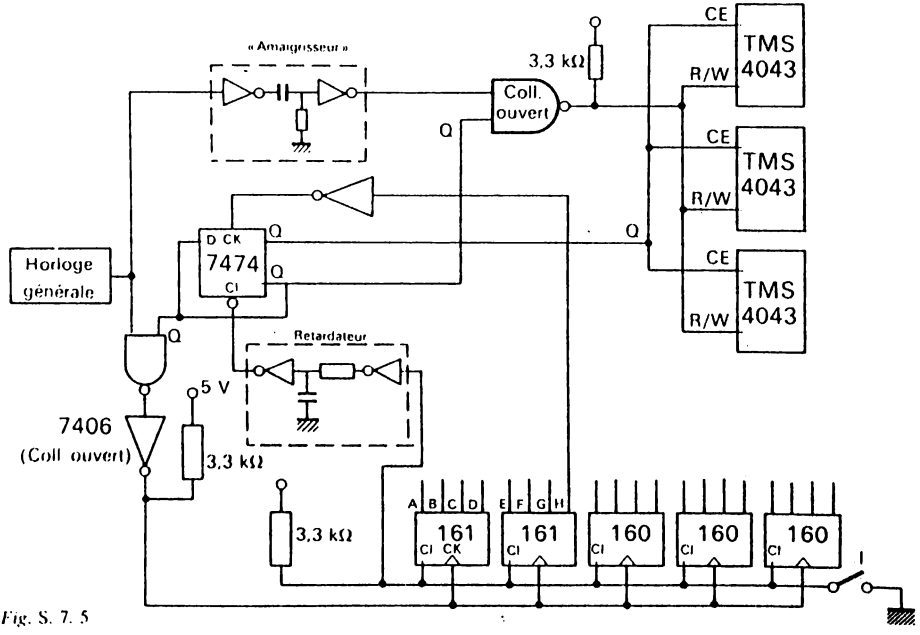


Fig. S. 7. 5

Comme, de plus, il n'est pas souhaitable que des impulsions d'écriture parviennent sur les mémoires après la fin du cycle, nous profiterons de la même sortie Q de cette bascule pour inhiber le signal W, tandis que la sortie Q, passant à 1, supprimera l'autorisation CE des mémoires.

Utilisation des mémoires en transcodage : Les mémoires ayant reçu leur contenu, on pourra retirer tous les éléments du montage d'écriture pour peu qu'on ait réalisé ce circuit sur une plaque de montage rapide.

La seule précaution, indispensable, sera de maintenir en permanence l'alimentation des mémoires, sinon leur contenu se volatiserait.

On peut ensuite câbler le circuit de transcodage.

Si, par contre, on veut réaliser un montage en circuit imprimé, les deux circuits d'écriture d'une part, d'exploitation des mémoires d'autre part, doivent pouvoir coexister. Il faudra aménager dans ce cas des groupes de portes « trois-états » sur les circuits adresses (binaire) et BCD de sortie des compteurs, isolant ces compteurs lorsque le cycle d'écriture est terminé.

On pourra alors couper l'alimentation à condition qu'avant toute phase d'utilisation des mémoires on remette en route le cycle d'écriture. Ce cycle peut fonctionner avec une horloge à 10 000 Hz et sa durée dans ce cas, sera de quelques dizaines de millisecondes.

Chapitre 11

3. Tableau de correspondance hexadécimal-décimal.

Voici le programme (en mnémoniques) que nous proposons :

(1) LDX # 00	X contiendra la valeur décimale.
(2) LDY # 00	Y sera l'index hexadécimal d'adressage.
(3) TXA	
(4) STA 9000, Y	(pour une meilleure compréhension nous n'inversons pas les octets de l'adresse).
(5) INY	
(6) CPY # 64	dès que Y > 63 le programme doit s'arrêter.
(7) BEQ (19)	renvoie au « stop » de fin de programme.
(8) INX	X est incrémenté, mais...
(9) TXA	il faut vérifier qu'il n'atteint pas une valeur commençant par A (1010_2).
(10) AND # 0F	on ne conserve que l'octet de poids faible
(11) CMP # 0A	et on compare à 1010_2
(12) BEQ (14)	tant que cette valeur n'est pas atteinte...
(13) JMP (3)	nous rebouclons
(14) TXA	sinon il faut ajouter 6 à la valeur de X
(15) CLC	
(16) ADC # 6	
(17) TAX	ceci étant fait...
(18) JMP (3)	on peut reboucler
(19) BRK	
(20) BRK	
(21) BRK	

4. Recherche de la valeur hexadécimale d'un nombre décimal compris entre 0 et 99.

Ce programme va balayer le contenu des mémoires 9000 à 9063 remplies par le programme précédent. Dès qu'il trouvera un nombre identique à la valeur décimale proposée, il doit s'arrêter et indiquer l'octet de poids faible de l'adresse.

On supposera que l'on range le nombre décimal dont on recherche la correspondance en mémoire 9070 et la solution en 9071.

Voici le programme proposé :

(1) LDY # 00	Y sera l'index d'adressage
(2) LDA 9000, Y	
(3) AND 9070	cette opération ne conserve le contenu de l'accumulateur que s'il est identique à celui de la mémoire 9070
(4) CMP 9070	est-ce bien le cas?
(5) BEQ (8)	si oui, on ira à l'instruction de rangement de la solution
(6) INY	sinon, poursuivre la recherche...
(7) JMP (2)	... en rebouclant
(8) STY 9071	ranger la solution en 9071
(9) BRK - BRK - BRK	et s'arrêter

Chapitre 13

1.

a) A l'étape d'attente 0 l'ensemble des conditions nécessaires à la marche du cycle est formé du « ET » de toutes les conditions suivantes :

MCY : appui sur le bouton-poussoir de marche du cycle
T1V : trémie 1 vide
T2V : trémie 2 vide
VF1 : vidange de la trémie 1 fermée
VF2 : vidange de la trémie 2 fermée
MAV : malaxeur vide
VFA : vidange du malaxeur fermée

Ce qui permet d'écrire la réceptivité associée à l'étape 0 et permettant le passage à l'étape suivante :

MCY . T1V . T2V . VF1 . VF2 . MAV . VFA

b) Deux séquences doivent être ensuite activées simultanément :

– la séquence 1 relative à la trémie 1,
– la séquence 2 relative à la trémie 2.

On a donc une « divergence en ET » à partir de l'étape 0.

c) **Séquence 1 :**

Étape 1 : remplissage de la trémie 1 (RT1).
Réceptivité : trémie 1 pleine (T1P)

– *Étape* 2 : vidange de la trémie 1 dans le malaxeur (VT1) et mise en route du malaxeur (MAL).
Réceptivité : trémie 1 vide (T1V)

– *Étape* 3 : malaxage (MAL), armement de la temporisation de 10 secondes.
Réceptivité : temporisation 10 s terminée.

d) **Séquence 2 :**

– *Étape* 7 : remplissage de la trémie 2 (RT2); armement de la temporisation de 3 s pour commencer le chauffage.

Réceptivité : temporisation 3 s terminée

– *Étape* 8 : remplissage de la trémie 2 (RT2). Chauffage de la trémie 2 (CHA) sous contrôle du thermostat (THE) (le chauffage n'est effectué que lorsque THE = 0, lorsque THE = 1 le chauffage s'arrête).

Réceptivité : trémie 2 pleine.

Étape 9 : chauffage sous contrôle du thermostat (CHA, THE).

Réceptivité : trémie 2 chaude (THE)

e) Lorsque les séquences 1 et 2 sont toutes deux terminées (convergence en ET), la suite de l'automatisme comporte :

– *Étape* 4 : vidange de la trémie 2 (VT2) et malaxage (MAL).

Réceptivité : trémie 2 vide (T2V)

– *Étape* 5 : malaxage (MAL). Armement de la temporisation de 15 secondes.

Réceptivité : temporisation 15 s terminée

– *Étape* 6 : vidange du malaxeur (VMA).

Réceptivité : malaxeur vide (MAV) permettant le retour à l'étape initiale 0.

Imprimerie GAUTHIER-VILLARS, France
Dépôt légal, Imprimeur, n° 3214
Dépôt légal : octobre 1989 *Imprimé en France*
Dépôt légal 1re édition : 1er trimestre 1986

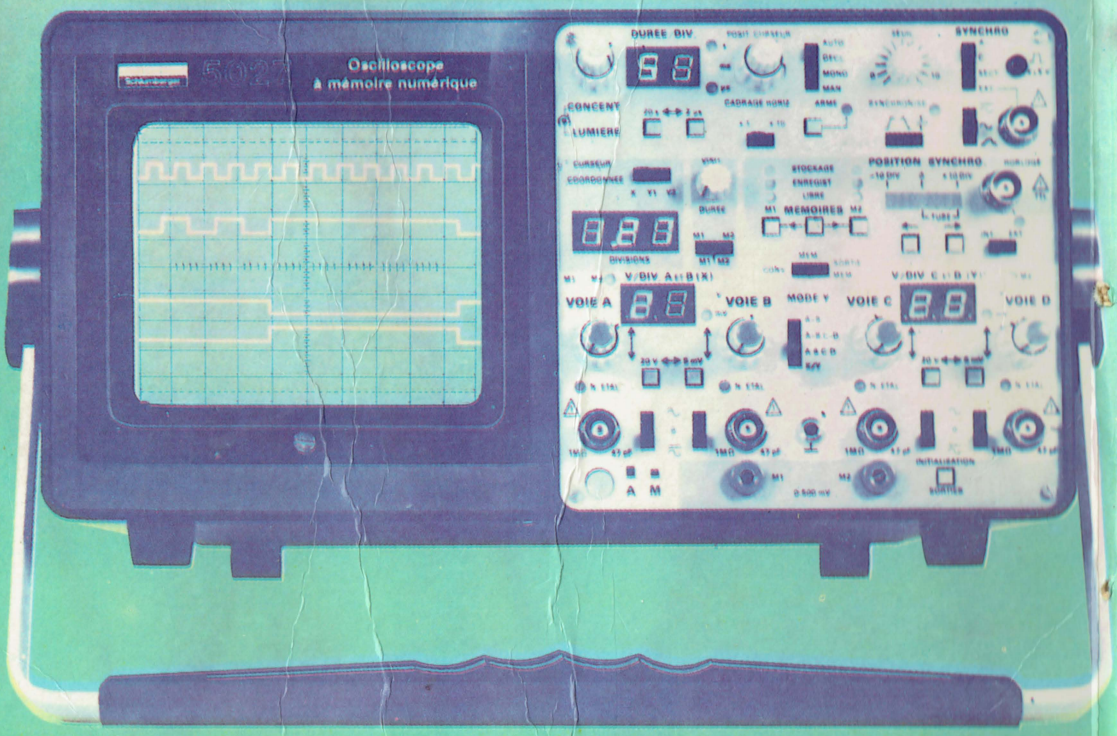
Présentation de "L'ÉLECTRONIQUE DIGITALE"

Les champs d'application de l'électronique digitale (ou numérique) sont infinis. De très nombreuses fonctions assurées jusqu'ici par des mécanismes complexes, coûteux et fragiles seront bientôt remplacés en grande partie par de petits circuits d'électronique fiables, peu coûteux et beaucoup plus performants.

Parallèlement, tout concepteur de produit, tout spécialiste de méthodes de fabrication, tout inventeur, du plus modeste au plus génial, se doit de connaître ces nouvelles techniques.

Ce livre cherche à en faciliter l'approche : à partir de la seule notion de transistor fonctionnant en commutation (saturé ou bloqué), il développe progressivement toutes les notions fondamentales de cette électronique des temps modernes, ainsi que tous les outils de raisonnement. Il présente, au fur et à mesure, les produits proposés par les fabricants de circuits intégrés et leur emploi.

Sa démarche claire et progressive, son style simple, le rendent accessible à toute personne désirant acquérir une solide connaissance de ces techniques, même avec un bagage de départ limité.



9 782040 159412

ISBN : 2-04-015941-X

P. Cabanis

Electronique digitale

Dunod